```
DDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBB      UUU         UUU     GGGGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUU         UUU     GGGGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUU         UUU     GGGGGGGGGGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEEEEEEEEEE      BBBBBBBBBBBB     UUU         UUU GGG
DDD         DDD EEEEEEEEEEE      BBBBBBBBBBBB     UUU         UUU GGG
DDD         DDD EEEEEEEEEEE      BBBBBBBBBBBB     UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG    GGGGGGGGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG    GGGGGGGGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG    GGGGGGGGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG          GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG          GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG          GGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUUUUUUUUUUUUUU     GGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUUUUUUUUUUUUUU     GGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUUUUUUUUUUUUUU     GGGGGGGGG
```

FILEID**DBGEVENT

```
;     1          0001  0 MODULE DBGEVENT (IDENT = 'V04-000') =
;     2          0002  0
;     3          0003  1 BEGIN
;     4          0004  1
;     5          0005  1 !**************************************************************************
;     6          0006  1 !*                                                                        *
;     7          0007  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                              *
;     8          0008  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                *
;     9          0009  1 !*  ALL RIGHTS RESERVED.                                                  *
;    10          0010  1 !*                                                                        *
;    11          0011  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
;    12          0012  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    13          0013  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;    14          0014  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;    15          0015  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;    16          0016  1 !*  TRANSFERRED.                                                           *
;    17          0017  1 !*                                                                        *
;    18          0018  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;    19          0019  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;    20          0020  1 !*  CORPORATION.                                                           *
;    21          0021  1 !*                                                                        *
;    22          0022  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
;    23          0023  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                *
;    24          0024  1 !*                                                                        *
;    25          0025  1 !*                                                                        *
;    26          0026  1 !**************************************************************************
;    27          0027  1 !
;    28          0028  1 ! WRITTEN BY
;    29          0029  1 !       Sidney R Maxwell III    May, 1982
;    30          0030  1 !
;    31          0031  1 ! MODULE FUNCTION
;    32          0032  1 !       This module contains the support routines for the new event-points.
;    33          0033  1 !       Included are the syntax and semantics routines for processing the
;    34          0034  1 !       new command forms, and the support for the new data structures.
;    35          0035  1 !
;    36          0036  1
;    37          0037  1 REQUIRE 'SRC$:DBGPROLOG.REQ';
;    38          0171  1
;    39          0172  1 LIBRARY 'LIB$:DBGGEN.L32';
;    40          0173  1
;    41          0174  1 FORWARD ROUTINE
;    42          0175  1     DBG$EVENT_INITIALIZATION : NOVALUE,  ! Initialize the event structures
;    43          0176  1     DBG$EVENT_SYNTAX,                    ! Handle BREAK/TRACE/WATCH syntax
;    44          0177  1     DBG$EVENT_SEMANTICS,                 ! Handle BREAK/TRACE/WATCH semantics
;    45          0178  1     DBG$EVENT_SHOW_CANCEL_SYNTAX,        ! [SHOW|CANCEL] BREAK/TRACE/WATCH syntax
;    46          0179  1     DBG$EVENT_SHOW_CANCEL_SEMANTICS,     ! Handle BREAK/TRACE/WATCH semantics
;    47          0180  1     DBG$EVENT_CANCEL_ALL: NOVALUE,       ! Cancel all eventpoints
;    48          0181  1     DBG$ACTIVATE_EVENTS: NOVALUE,        ! Activate all set events
;    49          0182  1     DBG$ACTIVATE_EVENT: NOVALUE,         ! Activate a set event
;    50          0183  1     DBG$DEACTIVATE_EVENTS: NOVALUE,      ! Deactivate all set events
;    51          0184  1     DBG$DEACTIVATE_EVENT: NOVALUE,       ! Deactivate a set event
;    52          0185  1     DBG$EXCEPTION_HANDLER,               ! Event Exception Handler
;    53          0186  1     DBG$PROCESS_EVENT,                   ! Process an event
;    54          0187  1     DBG$UPDATE_WATCHPOINTS: NOVALUE,     ! Update the current value of each watchpoint
;    55          0188  1     ANNOUNCE_EVENT: NOVALUE,             ! Announce an event
;    56          0189  1     INSQUE1: NOVALUE,                    ! Insert a queue entry (1st vs 0th)
;    57          0190  1     REMQUE1: NOVALUE,                    ! Remove a queue entry (1st vs 0th)
```

```
;    58          0191  1     SHOW_EVENT_ENTRY: NOVALUE,          ! Show an Event Entry
;    59          0192  1     DELETE_EVENT_ENTRY: NOVALUE,        ! Delete an Event Entry (et al)
;    60          0193  1     PRINT_SOURCE: NOVALUE,              ! Print a source line
;    61          0194  1     PRINT_PC_AND_INSTRUCTION: NOVALUE,  ! Print PC and instruction
;    62          0195  1     PRINT_PC_INST_HANDLER,              ! Handler for above
;    63          0196  1     COMPARE_VMSDESC,                    ! ???
;    64          0197  1     GET_WATCH_POINT_VALUE: NOVALUE,     ! Get the value of the watch pointing
;    65          0198  1                                         !       variable
;    66          0199  1     PRIM_TO_VAL: NOVALUE,               ! Local cover routine for DBG$PRIM_TO_VAL
;    67          0200  1     PRIM_TO_VAL_HANDLER,                ! Handler for above
;    68          0201  1     PARSE_WHEN_CONDITION: NOVALUE,      ! Handler to catch the bad status
;    69          0202  1                                         !       from parsing the WHEN clause
;    70          0203  1     MATCH_RIGHT_CALL_FRAME;             ! Make sure the call frame we got is
;    71          0204  1                                         !       the one we set bit 15 in saved
;    72          0205  1                                         !       PSW
```

```
;    74          0206  1 EXTERNAL ROUTINE
;    75          0207  1     DBG$NGET_ADDRESS,                  ! Get and address from a primary
;    76          0208  1     DBG$MAKE_VAL_DESC,                 ! VMS desc to valdesc
;    77          0209  1     DBG$PRIM_TO_VAL,                   ! Get a value descriptor from a primary
;    78          0210  1     DBG$NGET_SYMID,                    ! Get a SYMID
;    79          0211  1     DBG$NCOPY_DESC,                    ! Copy a primary descriptor
;    80          0212  1     DBG$NFREE_DESC,                    ! Free a primary descriptor
;    81          0213  1     DBG$STA_LOCK_SYMID,                ! Lock a SYMID
;    82          0214  1     DBG$STA_UNLOCK_SYMID,              ! Unlock a SYMID
;    83          0215  1     DBG$PRINT,                         ! 'Print' an FAO string
;    84          0216  1     DBG$NEWLINE,                       ! Dump 'printed' buffer
;    85          0217  1     DBG$PRINT_VALUE,                   ! 'Print' a value from a val desc
;    86          0218  1     DBG$PRINT_IDENTIFIER,              ! 'Print' an identifier from a primary
;    87          0219  1     DBG$PRINT_IDENTIFIER_PC,           ! 'Print' an identifier from an address
;    88          0220  1     DBG$PUTMSG,                        ! Put an exception message
;    89          0221  1     DBG$PC_TO_LINE_LOOKUP,             ! Match a PC to line
;    90          0222  1     DBG$SRC_TYPE_PC_SOURCE: NOVALUE,   ! Display source line
;    91          0223  1     DBG$INS_DECODE,                    ! Decode [and display] instruction
;    92          0224  1     DBG$OPCODE_INDEX,                  ! Get opcode from mnemonic
;    93          0225  1     DBG$FINAL_HANDL,                   ! DEBUG's final handler
;    94          0226  1     DBG$READ_ACCESS,                   ! Check for read access
;    95          0227  1     DBG$MAP_TO_REG_ADDR,               ! Map an address to register space
;    96          0228  1     DBG$NGET_PAGES,                    ! Get pages from a primary
;    97          0229  1     DBG$DATA_LENGTH,                   ! Get data bit length from VMS descriptor
;    98          0230  1     DBG$IS_IT_ENTRY,                   ! Checks address for entry point
;    99          0231  1     DBG$NMAKE_ARG_VECT,                ! Constructs a message argument vector
;   100          0232  1     DBG$NSAVE_DECIMAL_INTEGER,         ! Saves an inline numeric literal
;   101          0233  1     DBG$NSAVE_BREAK_BUFFER: NOVALUE,   ! Saves away a break action buffer
;   102          0234  1     DBG$GET_MEMORY,                    ! Get a block of permanent memory
;   103          0235  1     DBG$REL_MEMORY,                    ! Release a block of permanent memory
;   104          0236  1     DBG$GET_TEMPMEM,                   ! Get a block of temporary memory
;   105          0237  1     DBG$REL_TEMPMEM,                   ! Release all of temporary memory
;   106          0238  1     DBG$RST_TEMP_RELEASE: NOVALUE,     ! Release all temporary RST entries
;   107          0239  1     DBG$NNEXT_WORD,                    ! Isolates next word for syntax errors
;   108          0240  1     DBG$NMATCH,                        ! Routine to match keywords
;   109          0241  1     DBG$NPARSE_ADDRESS,                ! Interface routine to Address Expression Interpreter
;   110          0242  1     DBG$NPARSE_EXPRESSION,             ! Prase an expression
;   111          0243  1     DBG$NTYPE_CONV,                    ! Type converter
;   112          0244  1     DBG$WRITE_MEM,                     ! Write into memory
;   113          0245  1     DBG$NCIS_ADD,                      ! Add command string
;   114          0246  1     DBG$COMMAND_PROC,                  ! Execute a command list
;   115          0247  1     DBG$NINITIALIZE,                   ! Initialize for user commands
;   116          0248  1     DBG$NSYNTAX_ERROR,                 ! Constructs a syntax error message
;   117          0249  1     DBG$SET_STP_LVL : NOVALUE,         ! Set step level
;   118          0250  1     DBG$PSEUDO_PROG,                   ! DEBUG's call code
;   119          0251  1     LIB$SIGNAL: ADDRESSING_MODE (GENERAL),
;   120          0252  1     LIB$STOP;
;   121          0253  1
;   122          0254  1 EXTERNAL
;   123          0255  1     DBG$GB_CALL_NORMAL_RET: BYTE,      ! CALL command status flag; 0 = not in
;   124          0256  1                                        !       a CALL command, 1 = in a CALL
;   125          0257  1                                        !       that has not yet returned, 2 =
;   126          0258  1                                        !       in normal return from a CALL cmd
;   127          0259  1     DBG$GB_RADIX: VECTOR[3, BYTE],     ! Radix settings
;   128          0260  1     DBG$GB_LANGUAGE: BYTE,             ! Current set lang
;   129          0261  1     DBG$GL_SIGN_FLAG,                  ! Flag set to print '+' before signed
;   130          0262  1                                        !       variable
```

```
;  131     0263  1      DBG$GV_CONTROL: DBG$CONTROL_FLAGS,       ! Major State of DEBUG
;  132     0264  1      DBG$GB_STP_PTR:                          ! Current stepping modes
;  133     0265  1          REF EVENT$STEPPING_DESCRIPTOR,
;  134     0266  1      DBG$GB_MOD_PTR: REF VECTOR[,BYTE],       ! Current mode flags
;  135     0267  1      DBG$GL_CISREAD,                          ! Head of command input stream
;  136     0268  1      DBG$GL_CIS_LEVELS,                       ! Number of levels of CIS nesting
;  137     0269  1      DBG$GL_LIB_SIGNAL_ADDR,                  ! Address of LIB$SIGNAL
;  138     0270  1      DBG$GL_LIB_STOP_ADDR,                    ! Address of LIB$STOP
;  139     0271  1      DBG$GL_STEP_NUM,                         ! Step count
;  140     0272  1      DBG$GB_TAKE_CMD: BYTE,                   ! User commands okay (no go etc.)
;  141     0273  1      DBG$RUNFRAME: BLOCK[,BYTE],              ! Run frame
;  142     0274  1      DBG$GB_EXC_BRE_FLAG: BYTE,               ! Encountered an exception break
;  143     0275  1      DBG$GB_GO_ARG_FLAG: BYTE,                ! GO command had an argument
;  144     0276  1      DBG$GB_NO_GLOBALS: BYTE,                 ! Don't convert to global symbol
;  145     0277  1      DBG$OPCODE_KIND_TABLE:                   ! ???
;  146     0278  1                  VECTOR [512,WORD],
;  147     0279  1      DBG$OPCODE_NAME_TABLE:                   ! ???
;  148     0280  1                  BLOCKVECTOR [,10,BYTE],
;  149     0281  1      DBG$PSEUDO_EXIT,                         ! Label in pseudo code for CALL
;  150     0282  1      DBG$PSEUDO_SSI,                          ! Label in pseudo code for
;  151     0283  1                                               !    DBG_SSI_ROUTINE
;  152     0284  1      DBG$GB_SET_SSI_CNT: BYTE,                ! Is watch pointing active?
;  153     0285  1      DBG$TERM_HANDLR,                         ! Label in termination handler
;  154     0286  1      DBG$USER_EXIT,                           ! Label in exit code
;  155     0287  1      PRIM_HANDL_2,                            ! Label in primary handler
;  156     0288  1      DBG$GL_OUTPRAB: BLOCK[,BYTE],            ! RAB for 'OUTPUT'
;  157     0289  1      DBG$GL_ORIG_COMMAND_PTR,                 ! Pointer to original command string
;  158     0290  1      DBG$GL_UPCASE_COMMAND_PTR: VECTOR[2];    ! Pointers to start and end
;  159     0291  1                                               !    of current command string
;  160     0292  1
;  161     0293  1
;  162     0294  1  GLOBAL
;  163     0295  1      DBG$GB_SET_BREAK_FLAG:                   ! Set to TRUE during processing
;  164     0296  1                  BYTE INITIAL(0),             !      of address in SET BREAK
;  165     0297  1      DBG$GB_SET_WATCH_FLAG:                   ! Set to TRUE during processing
;  166     0298  1                  BYTE INITIAL (0);            !      of System service cycle
;  167     0299  1                                               !      (CALL to RET) for watch point
;  168     0300  1                                               !      value comparison
;  169     0301  1
;  170     0302  1  OWN
;  171     0303  1      INVALID_FLAG,                            ! Set to TRUE when a Primary becomes
;  172     0304  1                                               !    invalid (PRIM_TO_VAL_HANDLER)
;  173     0305  1      NEWVALUE: REF DBG$VALDESC,               ! New value for watch point variable
;  174     0306  1      OLDVALUE: REF DBG$VALDESC,               ! Old value for watch point variable
;  175     0307  1      TYPE_SOURCE,                             ! It's okay to type source
;  176     0308  1      ACCVIO_PC,                               ! PC at an ACCVIO
;  177     0309  1      WATCH_PC,                                ! PC at next instruction after System
;  178     0310  1                                               !      service call when watch pointing
;  179     0311  1                                               !      is trigged, used to report watch
;  180     0312  1                                               !      point address
;  181     0313  1      SKIP_ACCVIOS : INITIAL (FALSE),          ! We're [not initially] skipping ACCVIO's
;  182     0314  1      SKIP_WATCHES : INITIAL (FALSE),          ! Flag set to indicate We t-bit over RET
;  183     0315  1                                               !      of Service call, report watched
;  184     0316  1                                               !      value if there is one
;  185     0317  1      WHEN_CONDITION: REF DBG$VALDESC;         ! Condition for WHEN clause
;  186     0318  1
;  187     0319  1  BUILTIN
```

```
;  188          0320  1     INSQUE,          ! Insert an entry on a list
;  189          0321  1     REMQUE,          ! Remove an entry from a list
;  190          0322  1     PROBER;          ! Probe a location for read-access
```

```
;  192          0323  1 ! Define the QUEUE_HEAD data structure, used to reference the head of
;  193          0324  1 ! a queue.
;  194          0325  1 !
;  195          0326  1 FIELD
;  196          0327  1     QUEUE_HEAD_FIELDS =
;  197          0328  1         SET
;  198          0329  1         A_QUEUE_ENTRY = [0,A_],     ! Queue entry's address
;  199          0330  1
;  200          0331  1         L_QUEUE_FLINK = [0,L_],     ! Queue's 0th (normal) Forward LINK
;  201          0332  1         L_QUEUE_BLINK = [1,L_],     ! Queue's 0th (normal) Backward Link
;  202          0333  1
;  203          0334  1         L_QUEUE_FLINK1 = [2,L_],    ! Queue's 1st Forward LINK
;  204          0335  1         L_QUEUE_BLINK1 = [3,L_]     ! Queue's 1st Backward Link
;  205          0336  1         TES;
;  206          0337  1
;  207          0338  1
;  208          0339  1 MACRO QUEUE_HEAD = BLOCK [4,LONG] FIELD (QUEUE_HEAD_FIELDS)%;
;  209          0340  1
;  210          0341  1 GLOBAL
;  211          0342  1     EVENT$CMD_QUEUE:        QUEUE_HEAD,         ! Command Queue
;  212          0343  1     EVENT$DO_LIST_QUEUE:    QUEUE_HEAD,         ! DO List Queue
;  213          0344  1     EVENT$WHEN_QUEUE:       QUEUE_HEAD,         ! WHEN    Queue
;  214          0345  1     EVENT$PAGE_QUEUE:       QUEUE_HEAD,         ! PAGE    Queue
;  215          0346  1
;  216          0347  1
;  217          0348  1     ! The following is used to flag a stop, such as from a BREAK or the
;  218          0349  1     ! end of a STEP, as opposed to a TRACE....
;  219          0350  1     !
;  220          0351  1     EVENT$B_STOP_USER :     BYTE;
```

```
;   222     0352  1 OWN
;   223     0353  1     ! Opcode Table for /INSTRUCTION = (<mnemonic> ,...)
;   224     0354  1     !
;   225     0355  1     DBG$OPCODES_USER :  BITVECTOR[512],
;   226     0356  1
;   227     0357  1
;   228     0358  1     ! Opcode Table for STEP/INSTRUCTION = (<mnemonic> ,...)
;   229     0359  1     !
;   230     0360  1     DBG$OPCODES_STEP_USER : BITVECTOR[512],
;   231     0361  1
;   232     0362  1
;   233     0363  1     ! Opcode Table for /CALL events.
;   234     0364  1     !
;   235     0365  1     DBG$OPCODES_CALL :  BITVECTOR[512]
;   236     0366  1                         PRESET
;   237     0367  1                         ([%X'0FB'] = 1, ! CALLS
;   238     0368  1                          [%X'0FA'] = 1, ! CALLG
;   239     0369  1                          [%X'030'] = 1, ! BSBW
;   240     0370  1                          [%X'010'] = 1, ! BSBB
;   241     0371  1                          [%X'016'] = 1, ! JSB
;   242     0372  1                          [%X'005'] = 1, ! RSB
;   243     0373  1                          [%X'004'] = 1  ! RET
;   244     0374  1                         ),
;   245     0375  1
;   246     0376  1
;   247     0377  1     ! Opcode Table for /BRANCH events.
;   248     0378  1     !
;   249     0379  1     DBG$OPCODES_BRANCH :        BITVECTOR[512]
;   250     0380  1                         PRESET
;   251     0381  1                         ([%X'012'] = 1, ! BNEQ or BNEQU
;   252     0382  1                          [%X'013'] = 1, ! BEQL or BEQLU
;   253     0383  1                          [%X'014'] = 1, ! BGTR
;   254     0384  1                          [%X'015'] = 1, ! BLEQ
;   255     0385  1                          [%X'018'] = 1, ! BGEQ
;   256     0386  1                          [%X'019'] = 1, ! BLSS
;   257     0387  1                          [%X'01A'] = 1, ! BGTRU
;   258     0388  1                          [%X'01B'] = 1, ! BLEQU
;   259     0389  1                          [%X'01C'] = 1, ! BVC
;   260     0390  1                          [%X'01D'] = 1, ! BVS
;   261     0391  1                          [%X'01E'] = 1, ! BGEQU or BCC
;   262     0392  1                          [%X'01F'] = 1, ! BLSSU or BCS
;   263     0393  1                          [%X'011'] = 1, ! BRB
;   264     0394  1                          [%X'031'] = 1, ! BRW
;   265     0395  1                          [%X'017'] = 1, ! JMP
;   266     0396  1                          [%X'0E0'] = 1, ! BBS
;   267     0397  1                          [%X'0E1'] = 1, ! BBC
;   268     0398  1                          [%X'0E2'] = 1, ! BBSS
;   269     0399  1                          [%X'0E3'] = 1, ! BBCS
;   270     0400  1                          [%X'0E4'] = 1, ! BBSC
;   271     0401  1                          [%X'0E5'] = 1, ! BBCC
;   272     0402  1                          [%X'0E6'] = 1, ! BBSSI
;   273     0403  1                          [%X'0E7'] = 1, ! BBCCI
;   274     0404  1                          [%X'0E8'] = 1, ! BLBS
;   275     0405  1                          [%X'0E9'] = 1, ! BLBC
;   276     0406  1                          [%X'09D'] = 1, ! ACBB
;   277     0407  1                          [%X'03D'] = 1, ! ACBW
;   278     0408  1                          [%X'0F1'] = 1, ! ACBL
```

```
;   279     0409  1                          [%X'04F'] = 1, ! ACBF
;   280     0410  1                          [%X'06F'] = 1, ! ACBD
;   281     0411  1                          [%X'14F'] = 1, ! ACBG
;   282     0412  1                          [%X'16F'] = 1, ! ACBH
;   283     0413  1                          [%X'0F3'] = 1, ! AOBLEQ
;   284     0414  1                          [%X'0F2'] = 1, ! AOBLSS
;   285     0415  1                          [%X'0F4'] = 1, ! SOBGEQ
;   286     0416  1                          [%X'0F5'] = 1, ! SOBGTR
;   287     0417  1                          [%X'08F'] = 1, ! CASEB
;   288     0418  1                          [%X'0AF'] = 1, ! CASEW
;   289     0419  1                          [%X'0CF'] = 1  ! CASEL
;   290     0420  1                         ),
;   291     0421  1
;   292     0422  1
;   293     0423  1      ! Opcode Table for stepping /OVER.
;   294     0424  1      !
;   295     0425  1      DBG$OPCODES_STEP_OVER : BITVECTOR[512]
;   296     0426  1                              PRESET
;   297     0427  1                              ([%X'0FB'] = 1, ! CALLS
;   298     0428  1                               [%X'0FA'] = 1, ! CALLG
;   299     0429  1                               [%X'030'] = 1, ! BSBW
;   300     0430  1                               [%X'010'] = 1, ! BSBB
;   301     0431  1                               [%X'016'] = 1  ! JSB
;   302     0432  1                              );
```

```
;  304          0433  1 MACRO
;  305          0434  1
;  306          0435  1     ! Calculate word length from bit length.
;  307          0436  1     !
;  308        M 0437  1     BIT_TO_WORD_LENGTH (BIT_LENGTH) =
;  309        M 0438  1         ((BIT_LENGTH + %BPVAL - 1) / %BPVAL)
;  310          0439  1     %,
;  311          0440  1
;  312          0441  1
;  313          0442  1     ! Setup a UPLIT %ASCIC string.
;  314          0443  1     !
;  315        M 0444  1     $AC (STRING) =
;  316        M 0445  1         UPLIT BYTE (%ASCIC STRING)
;  317          0446  1     %,
;  318          0447  1
;  319          0448  1
;  320          0449  1     ! Test <string> for a match with the input string.
;  321          0450  1     !
;  322        M 0451  1     MATCH (STRING) =
;  323        M 0452  1         (DBG$NMATCH(.INPUT_DESC, STRING,
;  324          0453  1                 %IF %NULL (%REMAINING) %THEN 1 %ELSE %REMAINING %FI)) %,
;  325          0454  1
;  326          0455  1
;  327          0456  1     ! Match <string> to the input string, and give a syntax
;  328          0457  1     ! error if <string> does not match.
;  329          0458  1     !
;  330        M 0459  1     NEED_MATCH (STRING) =
;  331          0460  1         (IF NOT MATCH (STRING) THEN SYNTAX_ERROR;) %,
;  332          0461  1
;  333          0462  1
;  334          0463  1     ! Syntax Error
;  335          0464  1     !
;  336        M 0465  1     SYNTAX_ERROR =
;  337        M 0466  1         BEGIN
;  338        M 0467  1         .MESSAGE_VECT =
;  339        M 0468  1             (IF MATCH (DBG$CS_CR)
;  340        M 0469  1              THEN
;  341        M 0470  1                 DBG$NMAKE_ARG_VECT (DBG$_NEEDMORE)
;  342        M 0471  1              ELSE
;  343        M 0472  1                 DBG$NSYNTAX_ERROR (DBG$NNEXT_WORD (.INPUT_DESC))
;  344        M 0473  1             );
;  345        M 0474  1             RETURN STS$K_SEVERE
;  346        M 0475  1         END
;  347          0476  1     %,
;  348          0477  1
;  349          0478  1
;  350          0479  1     ! Allocate an adverb node.
;  351          0480  1     !
;  352        M 0481  1     GET_ADVERB_NODE =
;  353        M 0482  1         BEGIN
;  354        M 0483  1         ADVERB_NODE = DBG$GET_TEMPMEM (DBG$K_ADVERB_NODE_SIZE);
;  355        M 0484  1         ADVERB_NODE [DBG$L_ADVERB_LINK] = 0;
;  356        M 0485  1         .ADVERB_LINK = .ADVERB_NODE;
;  357        M 0486  1         ADVERB_LINK = ADVERB_NODE [DBG$L_ADVERB_LINK]
;  358        M 0487  1         END
;  359          0488  1     %,
;  360          0489  1
```

```
;  361          0490  1
;  362          0491  1      ! Allocate a noun node.
;  363          0492  1      !
;  364        M 0493  1      GET_NOUN_NODE =
;  365        M 0494  1          BEGIN
;  366        M 0495  1          NOUN_NODE = DBG$GET_TEMPMEM (DBG$K_NOUN_NODE_SIZE);
;  367        M 0496  1          NOUN_NODE [DBG$L_NOUN_LINK] = 0;
;  368        M 0497  1          .NOUN_LINK = .NOUN_NODE;
;  369        M 0498  1          NOUN_LINK = NOUN_NODE [DBG$L_NOUN_LINK]
;  370        M 0499  1          END
;  371          0500  1      %.
;  372          0501  1
;  373          0502  1
;  374          0503  1      ! Write an opcode to memory.
;  375          0504  1      !
;  376        M 0505  1      WRITE_OPCODE (ADDRESS, OPCODE) =
;  377        M 0506  1          (   LOCAL
;  378        M 0507  1                  OP;
;  379        M 0508  1
;  380        M 0509  1              OP = OPCODE;
;  381        M 0510  1              DBG$WRITE_MEM (ADDRESS, OP, 1)
;  382        M 0511  1          )
;  383          0512  1      %;
```

```
;  385          0513  1 ! Literals
;  386          0514  1 !
;  387          0515  1 LITERAL
;  388          0516  1
;  389          0517  1
;  390          0518  1     ! Define the adverb literals, used to describe the contents
;  391          0519  1     ! of an adverb node.
;  392          0520  1     !
;  393        P 0521  1     ENUMERATE   (0, ADVERB_AFTER,
;  394        P 0522  1                     ADVERB_TEMPORARY,
;  395        P 0523  1                     ADVERB_SILENT,
;  396        P 0524  1                     ADVERB_NOSILENT,
;  397        P 0525  1                     ADVERB_WHEN,
;  398        P 0526  1                     ADVERB_DO,
;  399        P 0527  1
;  400        P 0528  1                     ADVERB_READ,
;  401        P 0529  1                     ADVERB_WRITE,
;  402        P 0530  1                     ADVERB_MODIFY,
;  403        P 0531  1                     ADVERB_EXECUTE,
;  404        P 0532  1                     ADVERB_RETURN,
;  405        P 0533  1                     ADVERB_EXCEPTION,
;  406        P 0534  1                     ADVERB_SOURCE,
;  407        P 0535  1                     ADVERB_NOSOURCE,
;  408        P 0536  1                     ADVERB_SYSTEM,
;  409        P 0537  1                     ADVERB_NOSYSTEM,
;  410        P 0538  1                     ADVERB_INTO,
;  411        P 0539  1                     ADVERB_OVER,
;  412        P 0540  1                     ADVERB_CALL,
;  413        P 0541  1                     ADVERB_BRANCH,
;  414        P 0542  1                     ADVERB_LINE,
;  415        P 0543  1                     ADVERB_INSTRUCTION,
;  416        P 0544  1                     ADVERB_OPCODE,
;  417        P 0545  1                     ADVERB_ALL
;  418          0546  1                 );
;  419          0547  1
```

```
;  421     0548  1 GLOBAL ROUTINE DBG$EVENT_INITIALIZATION: NOVALUE =
;  422     0549  1 !
;  423     0550  1 ! FUNCTION
;  424     0551  1 !       This routine is called to initialize the event structures when
;  425     0552  1 !       DEBUG is first entered.
;  426     0553  1 !
;  427     0554  1 ! INPUTS
;  428     0555  1 !       None
;  429     0556  1 !
;  430     0557  1 ! OUTPUTS
;  431     0558  1 !       None
;  432     0559  1 !
;  433     0560  1
;  434     0561  2     BEGIN
;  435     0562  2
;  436     0563  2
;  437     0564  2     ! Intialize the head of the Command Queue.
;  438     0565  2     !
;  439     0566  2     EVENT$CMD_QUEUE [L_QUEUE_FLINK] = EVENT$CMD_QUEUE;
;  440     0567  2     EVENT$CMD_QUEUE [L_QUEUE_BLINK] = EVENT$CMD_QUEUE;
;  441     0568  2     EVENT$CMD_QUEUE [L_QUEUE_FLINK1] = EVENT$CMD_QUEUE;
;  442     0569  2     EVENT$CMD_QUEUE [L_QUEUE_BLINK1] = EVENT$CMD_QUEUE;
;  443     0570  2
;  444     0571  2
;  445     0572  2     ! Initialize the head of the DO List Queue.
;  446     0573  2     !
;  447     0574  2     EVENT$DO_LIST_QUEUE [L_QUEUE_FLINK] = EVENT$DO_LIST_QUEUE;
;  448     0575  2     EVENT$DO_LIST_QUEUE [L_QUEUE_BLINK] = EVENT$DO_LIST_QUEUE;
;  449     0576  2
;  450     0577  2
;  451     0578  2     ! Initialize the head of the WHEN Queue.
;  452     0579  2     !
;  453     0580  2     EVENT$WHEN_QUEUE [L_QUEUE_FLINK] = EVENT$WHEN_QUEUE;
;  454     0581  2     EVENT$WHEN_QUEUE [L_QUEUE_BLINK] = EVENT$WHEN_QUEUE;
;  455     0582  2
;  456     0583  2
;  457     0584  2     ! Initialize the head of the PAGE Queue.
;  458     0585  2     !
;  459     0586  2     EVENT$PAGE_QUEUE [L_QUEUE_FLINK] = EVENT$PAGE_QUEUE;
;  460     0587  2     EVENT$PAGE_QUEUE [L_QUEUE_BLINK] = EVENT$PAGE_QUEUE;
;  461     0588  1     END;

                                .TITLE  DBGFVENT
                                .IDENT  \V04-000\

                                .PSECT  DBG$OWN,NOEXE,  PIC,2

                        00000 INVALID_FLAG:
                                .BLKB   4
                        00004 NEWVALUE:
                                .BLKB   4
                        00008 OLDVALUE:
                                .BLKB   4
                        0000C TYPE_SOURCE:
                                .BLKB   4
                        00010 ACCVIO_PC:
```

```
                                .BLKB   4
                00014 WATCH_PC:
                                .BLKB   4
            00000000  00018 SKIP_ACCVIOS:
                                .LONG   0                                       ;
            00000000  0001C SKIP_WATCHES:
                                .LONG   0                                       ;
                00020 WHEN_CONDITION:
                                .BLKB   4
                00024 DBG$OPCODES_USER:
                                .BLKB   64
                00064 DBG$OPCODES_STEP_USER:
                                .BLKB   64
                30  000A4 DBG$OPCODES_CALL:
                                .BYTE   48                                      ;
                00  000A5           .BYTE   0                                   ;
                41  000A6           .BYTE   65                                  ;
                00# 000A7           .BYTE   0[3]                                ;
                01  000AA           .BYTE   1                                   ;
                00# 000AB           .BYTE   0[24]                               ;
                0C  000C3           .BYTE   12                                  ;
                    000C4           .BLKB   32
                00# 000E4 DBG$OPCODES_BRANCH:
                                .BYTE   0[2]                                    ;
             FF BE  000E6           .BYTE   -66, -1                             ;
                00# 000E8           .BYTE   0[2]                                ;
             20 02  000EA           .BYTE   2, 32                               ;
                00  000EC           .BYTE   0                                   ;
                80  000ED           .BYTE   -128                                ;
                00# 000EE           .BYTE   0[3]                                ;
                80  000F1           .BYTE   -128                                ;
                00# 000F2           .BYTE   0[3]                                ;
                80  000F5           .BYTE   -128                                ;
                00  000F6           .BYTE   0                                   ;
                20  000F7           .BYTE   32                                  ;
                00  000F8           .BYTE   0                                   ;
                80  000F9           .BYTE   -128                                ;
                00# 000FA           .BYTE   0[3]                                ;
                80  000FD           .BYTE   -128                                ;
                00# 000FE           .BYTE   0[2]                                ;
          3E 03 FF  00100           .BYTE   -1, 3, 62                           ;
                00# 00103           .BYTE   0[10]                               ;
                80  0010D           .BYTE   -128                                ;
                00# 0010E           .BYTE   0[3]                                ;
                80  00111           .BYTE   -128                                ;
                    00112           .BLKB   18
                00# 00124 DBG$OPCODES_STEP_OVER:
                                .BYTE   0[2]                                    ;
                41  00126           .BYTE   65                                  ;
                00# 00127           .BYTE   0[3]                                ;
                01  0012A           .BYTE   1                                   ;
                00# 0012B           .BYTE   0[24]                               ;
                0C  00143           .BYTE   12                                  ;
                    00144           .BLKB   32

                                .PSECT  DBG$GLOBAL,NOEXE,  PIC,2
```

```
00  00000 DBG$GB_SET_BREAK_FLAG::
                .BYTE   0                                                       ;
00  00001 DBG$GB_SET_WATCH_FLAG::                                               ;
                .BYTE   0
        00002   .BLKB   2
        00004 EVENT$CMD_QUEUE::
                .BLKB   16
        00014 EVENT$DO_LIST_QUEUE::
                .BLKB   16
        00024 EVENT$WHEN_QUEUE::
                .BLKB   16
        00034 EVENT$PAGE_QUEUE::
                .BLKB   16
        00044 EVENT$B_STOP_USER::
                .BLKB   1

                .EXTRN  DBG$NGET_ADDRESS
                .EXTRN  DBG$MAKE_VAL_DESC
                .EXTRN  DBG$PRIM_TO_VAL
                .EXTRN  DBG$NGET_SYMID, DBG$NCOPY_DESC
                .EXTRN  DBG$NFREE_DESC, DBG$STA_LOCK_SYMID
                .EXTRN  DBG$STA_UNLOCK_SYMID
                .EXTRN  DBG$PRINT, DBG$NEWLINE
                .EXTRN  DBG$PRINT_VALUE
                .EXTRN  DBG$PRINT_IDENTIFIER
                .EXTRN  DBG$PRINT_IDENTIFIER_PC
                .EXTRN  DBG$PUTMSG, DBG$PC_TO_LINE_LOOKUP
                .EXTRN  DBG$SRC_TYPE_PC_SOURCE
                .EXTRN  DBG$INS_DECODE, DBG$OPCODE_INDEX
                .EXTRN  DBG$FINAL_HANDL
                .EXTRN  DBG$READ_ACCESS
                .EXTRN  DBG$MAP_TO_REG_ADDR
                .EXTRN  DBG$NGET_PAGES, DBG$DATA_LENGTH
                .EXTRN  DBG$IS_IT_ENTRY
                .EXTRN  DBG$NMAKE_ARG_VECT
                .EXTRN  DBG$NSAVE_DECIMAL_INTEGER
                .EXTRN  DBG$NSAVE_BREAK_BUFFER
                .EXTRN  DBG$GET_MEMORY, DBG$REL_MEMORY
                .EXTRN  DBG$GET_TEMPMEM
                .EXTRN  DBG$REL_TEMPMEM
                .EXTRN  DBG$RST_TEMP_RELEASE
                .EXTRN  DBG$NNEXT_WORD, DBG$NMATCH
                .EXTRN  DBG$NPARSE_ADDRESS
                .EXTRN  DBG$NPARSE_EXPRESSION
                .EXTRN  DBG$NTYPE_CONV, DBG$WRITE_MEM
                .EXTRN  DBG$NCIS_ADD, DBG$COMMAND_PROC
                .EXTRN  DBG$NINITIALIZE
                .EXTRN  DBG$NSYNTAX_ERROR
                .EXTRN  DBG$SET_STP_LVL
                .EXTRN  DBG$PSEUDO_PROG
                .EXTRN  LIB$SIGNAL, LIB$STOP
                .EXTRN  DBG$GB_CALL_NORMAL_RET
                .EXTRN  DBG$GB_RADIX, DBG$GB_LANGUAGE
                .EXTRN  DBG$GL_SIGN_FLAG
                .EXTRN  DBG$GV_CONTROL, DBG$GB_STP_PTR
                .EXTRN  DBG$GB_MOD_PTR, DBG$GL_CISREAD
                .EXTRN  DBG$GL_CIS_LEVELS
```

```
                                                .EXTRN  DBG$GL_LIB_SIGNAL_ADDR
                                                .EXTRN  DBG$GL_LIB_STOP_ADDR
                                                .EXTRN  DBG$GL_STEP_NUM
                                                .EXTRN  DBG$GB_TAKE_CMD
                                                .EXTRN  DBG$RUNFRAME, DBG$GB_EXC_BRE_FLAG
                                                .EXTRN  DBG$GB_GO_ARG_FLAG
                                                .EXTRN  DBG$GB_NO_GLOBALS
                                                .EXTRN  DBG$OPCODE_KIND_TABLE
                                                .EXTRN  DBG$OPCODE_NAME_TABLE
                                                .EXTRN  DBG$PSEUDO_EXIT
                                                .EXTRN  DBG$PSEUDO_SSI, DBG$GB_SET_SSI_CNT
                                                .EXTRN  DBG$TERM_HANDLR
                                                .EXTRN  DBG$USER_EXIT, PRIM_HANDL_2
                                                .EXTRN  DBG$GL_OUTPRAB, DBG$GL_ORIG_COMMAND_PTR
                                                .EXTRN  DBG$GL_UPCASE_COMMAND_PTR

                                                .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                    0004 00000  .ENTRY  DBG$EVENT_INITIALIZATION, Save R2              ; 0548
            52 00000000'     EF   9E 00002      MOVAB   EVENT$CMD_QUEUE, R2                           ;
            62               62   9E 00009      MOVAB   EVENT$CMD_QUEUE, EVENT$CMD_QUEUE              ; 0566
       04   A2               62   9E 0000C      MOVAB   EVENT$CMD_QUEUE, EVENT$CMD_QUEUE+4            ; 0567
       08   A2               62   9E 00010      MOVAB   EVENT$CMD_QUEUE, EVENT$CMD_QUEUE+8            ; 0568
       0C   A2               62   9E 00014      MOVAB   EVENT$CMD_QUEUE, EVENT$CMD_QUEUE+12           ; 0569
       10   A2        10     A2   9E 00018      MOVAB   EVENT$DO_LIST_QUEUE, EVENT$DO_LIST_QUEUE      ; 0574
       14   A2        10     A2   9E 0001D      MOVAB   EVENT$DO_LIST_QUEUE, EVENT$DO_LIST_QUEUE+4    ; 0575
       20   A2        20     A2   9E 00022      MOVAB   EVENT$WHEN_QUEUE, EVENT$WHEN_QUEUE            ; 0580
       24   A2        20     A2   9E 00027      MOVAB   EVENT$WHEN_QUEUE, EVENT$WHEN_QUEUE+4          ; 0581
       30   A2        30     A2   9E 0002C      MOVAB   EVENT$PAGE_QUEUE, EVENT$PAGE_QUEUE            ; 0586
       34   A2        30     A2   9E 00031      MOVAB   EVENT$PAGE_QUEUE, EVENT$PAGE_QUEUE+4          ; 0587
                                      04 00036  RET                                                   ; 0588

; Routine Size:  55 bytes,    Routine Base:  DBG$CODE + 0000
```

```
;  463          0589  1 GLOBAL ROUTINE DBG$EVENT_SYNTAX(INPUT_DESC, VERB_NODE, MESSAGE_VECT) =
;  464          0590  1 !
;  465          0591  1 !  FUNCTION
;  466          0592  1 !       This routine parses a step, break, trace, or watch point command,
;  467          0593  1 !       after the '[SET] STEP', 'SET BREAK', 'SET TRACE', or 'SET WATCH'
;  468          0594  1 !       has been parsed.  This routine is called from DBGNSET (or DBGNSTEP).
;  469          0595  1 !
;  470          0596  1 !       The following syntax is accepted:
;  471          0597  1 !
;  472          0598  1 !       SET [ BREAK ! TRACE ! WATCH ]
;  473          0599  1 !           [ /AFTER : <n> ]
;  474          0600  1 !           [ /TEMPORARY ]
;  475          0601  1 !           [ /[NO]SILENT ]
;  476          0602  1 !           <event>
;  477          0603  1 !           [ WHEN (<expression>) ]
;  478          0604  1 !           [ DO (<command list>) ]
;  479          0605  1 !
;  480          0606  1 !       If the event is a watchpoint, <event> is:
;  481          0607  1 !
;  482          0608  1 !       <address> [ , <address> ]*
;  483          0609  1 !
;  484          0610  1 !       otherwise, <event> is:
;  485          0611  1 !
;  486          0612  1 !       [ /READ ! /WRITE ! /MODIFY ! /EXECUTE ! /RETURN ]
;  487          0613  1 !           <address> [ , <address> ]* !
;  488          0614  1 !       [ /EXCEPTION ] !
;  489          0615  1 !       [ /[NO]SOURCE ! /[NO]SYSTEM ! /INTO ! /OVER ]*
;  490          0616  1 !           [  /CALLS ! /RETURN ! /BRANCH ! /LINE ! /INSTRUCTION
;  491          0617  1 !               [ = [ <mnemonic> ! ( <mnemonic> [ , <mnemonic> ] ) ]]
;  492          0618  1 !           ]
;  493          0619  1 !
;  494          0620  1 !       SET STEP
;  495          0621  1 !       [ [NO]SILENT
;  496          0622  1 !       [ [NO]SOURCE ! [NO]SYSTEM ! INTO ! OVER ]*
;  497          0623  1 !           [  CALLS ! RETURN ! BRANCH ! LINE ! INSTRUCTION
;  498          0624  1 !               [ = [ <mnemonic> ! ( <mnemonic> [ , <mnemonic> ] ) ]]
;  499          0625  1 !           ]
;  500          0626  1 !
;  501          0627  1 !       STEP
;  502          0628  1 !       [ /[NO]SILENT
;  503          0629  1 !       [ /[NO]SOURCE ! /[NO]SYSTEM ! /INTO ! /OVER ]*
;  504          0630  1 !           [  /CALLS ! /RETURN ! /BRANCH ! /LINE ! /INSTRUCTION
;  505          0631  1 !               [ = [ <mnemonic> ! ( <mnemonic> [ , <mnemonic> ] ) ]]
;  506          0632  1 !           ]
;  507          0633  1 !
;  508          0634  1 !
;  509          0635  1 ! INPUTS
;  510          0636  1 !       INPUT_DESC:     The current command line input descriptor.
;  511          0637  1 !       VERB_NODE:      A pointer to a DEBUG verb node, with the COMPOSITE
;  512          0638  1 !                       attribute set to EVENT$K_SET_BREAK, EVENT$K_SET_TRACE,
;  513          0639  1 !                       EVENT$K_SET_WATCH, EVENT$K_SET_STEP, or EVENT$K_STEP.
;  514          0640  1 !
;  515          0641  1 !       MESSAGE_VECT:   The address of a longword to contain the address
;  516          0642  1 !                       of a message argument vector.
;  517          0643  1 !
;  518          0644  1 ! OUTPUTS
;  519          0645  1 !       The parse tree headed by VERB_NODE is created to represent the
```

```
;   520          0646  1 !       command parsed.  Errors cause this routine to return STS$K_SEVERE,
;   521          0647  1 !       otherwise STS$K_SUCCESS.
;   522          0648  1 !
;   523          0649  1
;   524          0650  2     BEGIN
;   525          0651  2
;   526          0652  2     MAP
;   527          0653  2         VERB_NODE: REF DBG$VERB_NODE;   ! Verb node pointer
;   528          0654  2
;   529          0655  2     LOCAL
;   530          0656  2         STATUS,                         ! Returned status
;   531          0657  2         ADVERB_NODE: REF DBG$ADVERB_NODE, ! Adverb node pointer
;   532          0658  2         ADVERB_LINK: REF DBG$ADVERB_NODE, ! Adverb node link
;   533          0659  2         NOUN_NODE: REF DBG$NOUN_NODE,   ! Noun node pointer
;   534          0660  2         NOUN_LINK: REF DBG$NOUN_NODE,   ! Noun node link
;   535          0661  2         CMD_TYPE,                       ! Command type
;   536          0662  2         CMD_KIND,                       ! Command Kind
;   537          0663  2         SUB_KIND,                       ! Command sub-kind
;   538          0664  2         AFTER_COUNT,                    ! After count
;   539          0665  2         TEMPORARY,                      ! Temporary flag
;   540          0666  2         SILENT,                         ! Silent flag
;   541          0667  2         SOURCE,                         ! Source flag
;   542          0668  2         NOSILENT,                       ! Nosilent flag
;   543          0669  2         NOSOURCE,                       ! Nosource flag
;   544          0670  2         SYSTEM,                         ! System flag
;   545          0671  2         NOSYSTEM,                       ! Nosystem flag
;   546          0672  2         INTO,                           ! Into flag
;   547          0673  2         OVER;                           ! Over flag
;   548          0674  2
;   549          0675  2
;   550          0676  2     BIND
;   551          0677  2
;   552          0678  2
;   553          0679  2         ! Define the character strings to parse.
;   554          0680  2         !
;   555          0681  2         DBG$CS_AFTER =          $AC ('AFTER'),
;   556          0682  2         DBG$CS_TEMPORARY =      $AC ('TEMPORARY'),
;   557          0683  2         DBG$CS_SILENT =         $AC ('SILENT'),
;   558          0684  2         DBG$CS_NOSILENT =       $AC ('NOSILENT'),
;   559          0685  2         DBG$CS_WHEN =           $AC ('WHEN'),
;   560          0686  2         DBG$CS_DO =             $AC ('DO'),
;   561          0687  2         DBG$CS_READ =           $AC ('READ'),
;   562          0688  2         DBG$CS_WRITE =          $AC ('WRITE'),
;   563          0689  2         DBG$CS_MODIFY =         $AC ('MODIFY'),
;   564          0690  2         DBG$CS_EXECUTE =        $AC ('EXECUTE'),
;   565          0691  2         DBG$CS_RETURN =         $AC ('RETURN'),
;   566          0692  2         DBG$CS_SOURCE =         $AC ('SOURCE'),
;   567          0693  2         DBG$CS_NOSOURCE =       $AC ('NOSOURCE'),
;   568          0694  2         DBG$CS_SYSTEM =         $AC ('SYSTEM'),
;   569          0695  2         DBG$CS_NOSYSTEM =       $AC ('NOSYSTEM'),
;   570          0696  2         DBG$CS_INTO =           $AC ('INTO'),
;   571          0697  2         DBG$CS_OVER =           $AC ('OVER'),
;   572          0698  2         DBG$CS_EXCEPTION =      $AC ('EXCEPTION'),
;   573          0699  2         DBG$CS_CALL =           $AC ('CALLS'),
;   574          0700  2         DBG$CS_BRANCH =         $AC ('BRANCH'),
;   575          0701  2         DBG$CS_LINE =           $AC ('LINE'),
;   576          0702  2         DBG$CS_INSTRUCTION =    $AC ('INSTRUCTION'),
```

```
;  577          0703  2          DBG$CS_EQUALS =         $AC ('='),
;  578          0704  2          DBG$CS_L_PAREN =        $AC ('('),
;  579          0705  2          DBG$CS_R_PAREN =        $AC (')'),
;  580          0706  2          DBG$CS_COMMA =          $AC (','),
;  581          0707  2          DBG$CS_SLASH =          $AC ('/'),
;  582          0708  2          DBG$CS_COLON =          $AC (':'),
;  583          0709  2          DBG$CS_CR =             UPLIT BYTE (1, DBG$K_CAR_RETURN);
;  584          0710  2
;  585          0711  2
;  586          0712  2
;  587          0713  2      ! Initialize the verb's adverb and noun pointers to null.
;  588          0714  2      !
;  589          0715  2      VERB_NODE [DBG$L_VERB_ADVERB_PTR] = 0;
;  590          0716  2      VERB_NODE [DBG$L_VERB_OBJECT_PTR] = 0;
;  591          0717  2
;  592          0718  2
;  593          0719  2      ! Initialize the adverb and noun links.
;  594          0720  2      !
;  595          0721  2      ADVERB_LINK = VERB_NODE [DBG$L_VERB_ADVERB_PTR];
;  596          0722  2      NOUN_LINK = VERB_NODE [DBG$L_VERB_OBJECT_PTR];
;  597          0723  2
;  598          0724  2
;  599          0725  2      ! Set the default command descriptors....
;  600          0726  2      !
;  601          0727  2      SELECTONE .VERB_NODE [DBG$B_VERB_COMPOSITE] OF
;  602          0728  2          SET
;  603          0729  2
;  604          0730  2
;  605          0731  2          [EVENT$K_SET_BREAK]:
;  606          0732  2              BEGIN
;  607          0733  2              CMD_TYPE = EVENT$K_TYPE_BREAK;
;  608          0734  2              CMD_KIND = EVENT$K_KIND_ACC;
;  609          0735  2              SUB_KIND = EVENT$K_ACC_EXEC;
;  610          0736  2              END;
;  611          0737  2
;  612          0738  2
;  613          0739  2          [EVENT$K_SET_TRACE]:
;  614          0740  2              BEGIN
;  615          0741  2              CMD_TYPE = EVENT$K_TYPE_TRACE;
;  616          0742  2              CMD_KIND = EVENT$K_KIND_ACC;
;  617          0743  2              SUB_KIND = EVENT$K_ACC_EXEC;
;  618          0744  2              END;
;  619          0745  2
;  620          0746  2
;  621          0747  2          [EVENT$K_SET_WATCH]:
;  622          0748  2              BEGIN
;  623          0749  2              CMD_TYPE = EVENT$K_TYPE_WATCH;
;  624          0750  2              CMD_KIND = EVENT$K_KIND_ACC;
;  625          0751  2              SUB_KIND = EVENT$K_ACC_MDFY;
;  626          0752  2              END;
;  627          0753  2
;  628          0754  2
;  629          0755  2          [EVENT$K_SET_STEP,EVENT$K_STEP]:
;  630          0756  2              BEGIN
;  631          0757  2              CMD_TYPE = EVENT$K_TYPE_STEPS;
;  632          0758  2              SUB_KIND = .DBG$GB_STP_PTR [EVENT$B_STEPPING_KIND];
;  633          0759  2              IF .SUB_KIND EQLU EVENT$K_ACC_RTRN
```

```
;  634     0760  3                THEN
;  635     0761  3                    CMD_KIND = EVENT$K_KIND_ACC
;  636     0762  3
;  637     0763  3                ELSE IF .SUB_KIND EQLU EVENT$K_EXC_EXC
;  638     0764  3                THEN
;  639     0765  3                    CMD_KIND = EVENT$K_KIND_EXC
;  640     0766  3
;  641     0767  3                ELSE
;  642     0768  3                    CMD_KIND = EVENT$K_KIND_INS;
;  643     0769  3
;  644     0770  3                END;
;  645     0771  2            TES;
;  646     0772  2
;  647     0773  2
;  648     0774  2        ! Setup the other default values.
;  649     0775  2        !
;  650     0776  2        AFTER_COUNT = 1;
;  651     0777  2        TEMPORARY = FALSE;
;  652     0778  2        SILENT = FALSE;
;  653     0779  2        NOSILENT = FALSE;
;  654     0780  2        SOURCE = FALSE;
;  655     0781  2        NOSOURCE = FALSE;
;  656     0782  2        SYSTEM = FALSE;
;  657     0783  2        NOSYSTEM = FALSE;
;  658     0784  2        INTO = FALSE;
;  659     0785  2        OVER = FALSE;
;  660     0786  2        IF .VERB_NODE [DBG$B_VERB_COMPOSITE] EQL EVENT$K_STEP
;  661     0787  2        THEN
;  662     0788  2            SILENT = NOT .DBG$GB_STP_PTR[EVENT$V_STEPPING_NOSILENT];
;  663     0789  2
;  664     0790  2
;  665     0791  2        ! Parse the command.  If this is not a SET STEP command, meaning that it
;  666     0792  2        ! is SET BREAK, SET TRACE, SET WATCH, or STEP, we look for qualifiers with
;  667     0793  2        ! "/" preceeding.
;  668     0794  2        !
;  669     0795  2        IF .VERB_NODE [DBG$B_VERB_COMPOSITE] NEQU EVENT$K_SET_STEP
;  670     0796  2        THEN
;  671     0797  2            BEGIN
;  672     0798  2
;  673     0799  2
;  674     0800  2            ! Look for the qualifiers....
;  675     0801  2            !
;  676     0802  2            ! Parse the command according to the following syntax:
;  677     0803  2            !
;  678     0804  2            !       [ /AFTER : <n> ]
;  679     0805  2            !       [ /[NO]SILENT ]
;  680     0806  2            !       [ /TEMPORARY ]
;  681     0807  2            !       [ /READ ! /WRITE ! /MODIFY ! /EXECUTE ! /RETURN ]
;  682     0808  2            !           <address> [ , <address> ]*
;  683     0809  2            !       /EXCEPTION
;  684     0810  2            !       [ /[NO]SOURCE ! /[NO]SYSTEM ! /INTO ! /OVER ]
;  685     0811  2            !       /CALLS
;  686     0812  2            !       /RETURN
;  687     0813  2            !       /BRANCH
;  688     0814  2            !       /INSTRUCTION [ = [ <mnemonic> ! [( <mnemonic> [ , <mnemonic> ]* ) ]]
;  689     0815  2            !
;  690     0816  2
```

```
;  691          0817  3          NOSILENT = TRUE;
;  692          0818  3          WHILE MATCH(DBG$CS_SLASH) DO
;  693          0819  4              BEGIN
;  694          0820  4              SELECTONE TRUE OF
;  695          0821  4                  SET
;  696          0822  4
;  697          0823  4
;  698          0824  4                  ! Parse the /AFTER : <n> qualifier.
;  699          0825  4                  !
;  700          0826  4                  [MATCH (DBG$CS_AFTER)]:
;  701          0827  5                      BEGIN
;  702          0828  5                      IF .CMD_TYPE EQLU EVENT$K_TYPE_BREAK OR
;  703          0829  5                         .CMD_TYPE EQLU EVENT$K_TYPE_TRACE OR
;  704          0830  5                         .CMD_TYPE EQLU EVENT$K_TYPE_WATCH
;  705          0831  5                      THEN
;  706          0832  6                          BEGIN
;  707          0833  6
;  708          0834  6
;  709          0835  6                          ! Match the ':'.
;  710          0836  6                          !
;  711          0837  6                          NEED_MATCH (DBG$CS_COLON);
;  712          0838  6
;  713          0839  6
;  714          0840  6                          ! Fill in the details.
;  715          0841  6                          !
;  716          0842  6                          IF NOT DBG$NSAVE_DECIMAL_INTEGER
;  717          0843  6                                 (.INPUT_DESC,
;  718          0844  6                                  AFTER_COUNT,
;  719          0845  6                                  .MESSAGE_VECT
;  720          0846  6                                 )
;  721          0847  6                          THEN
;  722          0848  6                              RETURN STS$K_SEVERE;
;  723          0849  6
;  724          0850  6                          END
;  725          0851  6
;  726          0852  5                      ELSE
;  727          0853  5                          SYNTAX_ERROR;
;  728          0854  5
;  729          0855  4                      END;
;  730          0856  4
;  731          0857  4
;  732          0858  4                  ! Parse the /TEMPORARY qualifier.
;  733          0859  4                  !
;  734          0860  4                  [MATCH (DBG$CS_TEMPORARY)]:
;  735          0861  5                      BEGIN
;  736          0862  5
;  737          0863  5
;  738          0864  5                      IF .CMD_TYPE EQLU EVENT$K_TYPE_BREAK OR
;  739          0865  5                         .CMD_TYPE EQLU EVENT$K_TYPE_TRACE OR
;  740          0866  5                         .CMD_TYPE EQLU EVENT$K_TYPE_WATCH
;  741          0867  5                      THEN
;  742          0868  6                          BEGIN
;  743          0869  6
;  744          0870  6
;  745          0871  6                          ! Fill in the details.
;  746          0872  6                          !
;  747          0873  6                          TEMPORARY = TRUE;
```

```
;  748     0874  6                    END
;  749     0875  6
;  750     0876  5                ELSE
;  751     0877  5                    SYNTAX_ERROR;
;  752     0878  5
;  753     0879  4                END;
;  754     0880  4
;  755     0881  4            ! Parse the /SILENT qualifier.
;  756     0882  4            !
;  757     0883  4            [MATCH (DBG$CS_SILENT, 2)]:
;  758     0884  4                BEGIN
;  759     0885  5                SILENT = TRUE;
;  760     0886  5                NOSILENT = FALSE;
;  761     0887  5                END;
;  762     0888  4
;  763     0889  4            ! Parse the /NOSILENT qualifier.
;  764     0890  4            !
;  765     0891  4            [MATCH (DBG$CS_NOSILENT, 4)]:
;  766     0892  4                BEGIN
;  767     0893  4                NOSILENT = TRUE;
;  768     0894  5                SILENT = FALSE;
;  769     0895  5                END;
;  770     0896  5
;  771     0897  4            ! Parse /READ.
;  772     0898  4            !
;  773     0899  4            [MATCH (DBG$CS_READ)]:
;  774     0900  4                BEGIN
;  775     0901  4
;  776     0902  4                ! Make sure that this is a SET BREAK or SET TRACE command,
;  777     0903  5                ! and then set the command kind to read access.
;  778     0904  5                !
;  779     0905  5                IF .CMD_TYPE EQLU EVENT$K_TYPE_BREAK OR
;  780     0906  5                   .CMD_TYPE EQLU EVENT$K_TYPE_TRACE
;  781     0907  5                THEN
;  782     0908  5                    BEGIN
;  783     0909  5                    CMD_KIND = EVENT$K_KIND_ACC;
;  784     0910  5                    SUB_KIND = EVENT$K_ACC_READ;
;  785     0911  5                    END
;  786     0912  6
;  787     0913  6                ELSE
;  788     0914  6                    SYNTAX_ERROR;
;  789     0915  6                END;
;  790     0916  6
;  791     0917  5            ! Parse /WRITE.
;  792     0918  5            !
;  793     0919  4            [MATCH (DBG$CS_WRITE)]:
;  794     0920  4                BEGIN
;  795     0921  4
;  796     0922  4                ! Make sure that this is a SET BREAK or SET TRACE command,
;  797     0923  4                ! and then set the command kind to write access.
;  798     0924  4                !
;  799     0925  5
;  800     0926  5
;  801     0927  5
;  802     0928  5
;  803     0929  5
;  804     0930  5
```

```
;   805     0931  5                 IF .CMD_TYPE EQLU EVENT$K_TYPE_BREAK OR
;   806     0932  5                    .CMD_TYPE EQLU EVENT$K_TYPE_TRACE
;   807     0933  5                 THEN
;   808     0934  6                     BEGIN
;   809     0935  6                     CMD_KIND = EVENT$K_KIND_ACC;
;   810     0936  6                     SUB_KIND = EVENT$K_ACC_WRIT;
;   811     0937  6                     END
;   812     0938  6
;   813     0939  5                 ELSE
;   814     0940  5                     SYNTAX_ERROR;
;   815     0941  4                 END;
;   816     0942  4
;   817     0943  4
;   818     0944  4             ! Parse /MODIFY.
;   819     0945  4             !
;   820     0946  4             [MATCH (DBG$CS_MODIFY)]:
;   821     0947  5                 BEGIN
;   822     0948  5
;   823     0949  5
;   824     0950  5                 ! Make sure that this is a SET BREAK or SET TRACE command,
;   825     0951  5                 ! and then set the command kind to modify access.
;   826     0952  5                 !
;   827     0953  5                 IF .CMD_TYPE EQLU EVENT$K_TYPE_BREAK OR
;   828     0954  5                    .CMD_TYPE EQLU EVENT$K_TYPE_TRACE
;   829     0955  5                 THEN
;   830     0956  6                     BEGIN
;   831     0957  6                     CMD_KIND = EVENT$K_KIND_ACC;
;   832     0958  6                     SUB_KIND = EVENT$K_ACC_MDFY;
;   833     0959  6                     END
;   834     0960  6
;   835     0961  5                 ELSE
;   836     0962  5                     SYNTAX_ERROR;
;   837     0963  5
;   838     0964  4                 END;
;   839     0965  4
;   840     0966  4
;   841     0967  4             ! Parse /EXECUTE <address> [ , <address> ]*.
;   842     0968  4             !
;   843     0969  4             [MATCH (DBG$CS_EXECUTE)]:
;   844     0970  5                 BEGIN
;   845     0971  5
;   846     0972  5
;   847     0973  5                 ! Make sure that this is a SET BREAK or SET TRACE command,
;   848     0974  5                 ! and then set the command kind to execute access.
;   849     0975  5                 !
;   850     0976  5                 IF .CMD_TYPE EQLU EVENT$K_TYPE_BREAK OR
;   851     0977  5                    .CMD_TYPE EQLU EVENT$K_TYPE_TRACE
;   852     0978  5                 THEN
;   853     0979  6                     BEGIN
;   854     0980  6                     CMD_KIND = EVENT$K_KIND_ACC;
;   855     0981  6                     SUB_KIND = EVENT$K_ACC_EXEC;
;   856     0982  6                     END
;   857     0983  6
;   858     0984  5                 ELSE
;   859     0985  5                     SYNTAX_ERROR;
;   860     0986  5
;   861     0987  4                 END;
```

```
;  862     0988  4
;  863     0989  4
;  864     0990  4                    ! Parse /RETURN <address> [ , <address> ]*.
;  865     0991  4                    !
;  866     0992  4                    [MATCH (DBG$CS_RETURN)]:
;  867     0993  5                        BEGIN
;  868     0994  5
;  869     0995  5
;  870     0996  5                        ! Make sure that this is not a SET WATCH command,
;  871     0997  5                        ! and then set the command kind to return access.
;  872     0998  5                        !
;  873     0999  5                        IF .CMD_TYPE NEQU EVENT$K_TYPE_WATCH
;  874     1000  5                        THEN
;  875     1001  6                            BEGIN
;  876     1002  6                            CMD_KIND = EVENT$K_KIND_ACC;
;  877     1003  6                            SUB_KIND = EVENT$K_ACC_RTRN;
;  878     1004  6                            END
;  879     1005  6
;  880     1006  5                        ELSE
;  881     1007  5                            SYNTAX_ERROR;
;  882     1008  5
;  883     1009  4                        END;
;  884     1010  4
;  885     1011  4
;  886     1012  4                    ! Parse /EXECEPTION.
;  887     1013  4                    !
;  888     1014  4                    [MATCH (DBG$CS_EXCEPTION)]:
;  889     1015  5                        BEGIN
;  890     1016  5
;  891     1017  5
;  892     1018  5                        ! Make sure that this is not a SET WATCH command,
;  893     1019  5                        ! and then set the command kind to execption.
;  894     1020  5                        !
;  895     1021  5                        IF .CMD_TYPE EQLU EVENT$K_TYPE_BREAK OR
;  896     1022  5                           .CMD_TYPE EQLU EVENT$K_TYPE_TRACE OR
;  897     1023  5                           .CMD_TYPE EQLU EVENT$K_TYPE_STEPS
;  898     1024  5                        THEN
;  899     1025  6                            BEGIN
;  900     1026  6                            CMD_KIND = EVENT$K_KIND_EXC;
;  901     1027  6                            SUB_KIND = EVENT$K_EXC_EXC;
;  902     1028  6                            END
;  903     1029  6
;  904     1030  5                        ELSE
;  905     1031  5                            SYNTAX_ERROR;
;  906     1032  5
;  907     1033  4                        END;
;  908     1034  4
;  909     1035  4
;  910     1036  4                    ! Parse /SOURCE.
;  911     1037  4                    !
;  912     1038  4                    [MATCH (DBG$CS_SOURCE, 2)]:
;  913     1039  5                        BEGIN
;  914     1040  5                        SOURCE = TRUE;
;  915     1041  5                        NOSOURCE = FALSE;
;  916     1042  4                        END;
;  917     1043  4
;  918     1044  4
```

```
;  919          1045  4                ! Parse /NOSOURCE.
;  920          1046  4                !
;  921          1047  4                [MATCH (DBG$CS_NOSOURCE, 4)]:
;  922          1048  5                    BEGIN
;  923          1049  5                    NOSOURCE = TRUE;
;  924          1050  5                    SOURCE = FALSE;
;  925          1051  4                    END;
;  926          1052  4
;  927          1053  4
;  928          1054  4                ! Parse /SYSTEM.
;  929          1055  4                !
;  930          1056  4                [MATCH (DBG$CS_SYSTEM, 2)]:
;  931          1057  5                    BEGIN
;  932          1058  5
;  933          1059  5
;  934          1060  5                    ! Make sure that this is not a SET WATCH command,
;  935          1061  5                    ! and then set the command kind to call.
;  936          1062  5                    !
;  937          1063  5                    IF .CMD_TYPE NEQU EVENT$K_TYPE_WATCH
;  938          1064  5                    THEN
;  939          1065  6                        BEGIN
;  940          1066  6                        SYSTEM = TRUE;
;  941          1067  6                        NOSYSTEM = FALSE;
;  942          1068  6                        END
;  943          1069  6
;  944          1070  5                    ELSE
;  945          1071  5                        SYNTAX_ERROR;
;  946          1072  5
;  947          1073  4                    END;
;  948          1074  4
;  949          1075  4
;  950          1076  4                ! Parse /NOSYSTEM.
;  951          1077  4                !
;  952          1078  4                [MATCH (DBG$CS_NOSYSTEM, 4)]:
;  953          1079  5                    BEGIN
;  954          1080  5
;  955          1081  5
;  956          1082  5                    ! Make sure that this is not a SET WATCH command,
;  957          1083  5                    ! and then set the command kind to call.
;  958          1084  5                    !
;  959          1085  5                    IF .CMD_TYPE NEQU EVENT$K_TYPE_WATCH
;  960          1086  5                    THEN
;  961          1087  6                        BEGIN
;  962          1088  6                        NOSYSTEM = TRUE;
;  963          1089  6                        SYSTEM = FALSE;
;  964          1090  6                        END
;  965          1091  6
;  966          1092  5                    ELSE
;  967          1093  5                        SYNTAX_ERROR;
;  968          1094  5
;  969          1095  4                    END;
;  970          1096  4
;  971          1097  4
;  972          1098  4                ! Parse /INTO.
;  973          1099  4                !
;  974          1100  4                [MATCH (DBG$CS_INTO)]:
;  975          1101  5                    BEGIN
```

```
;  976          1102  5
;  977          1103  5
;  978          1104  5                    ! Make sure that this is not a SET WATCH command,
;  979          1105  5                    ! and then set the command kind to call.
;  980          1106  5                    !
;  981          1107  5                    IF .CMD_TYPE NEQU EVENT$K_TYPE_WATCH
;  982          1108  5                    THEN
;  983          1109  6                        BEGIN
;  984          1110  6                        INTO = TRUE;
;  985          1111  6                        OVER = FALSE;
;  986          1112  6                        END
;  987          1113  6
;  988          1114  5                    ELSE
;  989          1115  5                        SYNTAX_ERROR;
;  990          1116  4                    END;
;  991          1117  4
;  992          1118  4                ! Parse /OVER.
;  993          1119  4                !
;  994          1120  4                [MATCH (DBG$CS_OVER)]:
;  995          1121  4                    BEGIN
;  996          1122  5
;  997          1123  5                    ! Make sure that this is not a SET WATCH command,
;  998          1124  5                    ! and then set the command kind to call.
;  999          1125  5                    !
; 1000          1126  5                    IF .CMD_TYPE NEQU EVENT$K_TYPE_WATCH
; 1001          1127  5                    THEN
; 1002          1128  5                        BEGIN
; 1003          1129  5                        OVER = TRUE;
; 1004          1130  6                        INTO = FALSE;
; 1005          1131  6                        END
; 1006          1132  6
; 1007          1133  6                    ELSE
; 1008          1134  6                        SYNTAX_ERROR;
; 1009          1135  5
; 1010          1136  5                    END;
; 1011          1137  5
; 1012          1138  4                ! Parse /CALLS.
; 1013          1139  4                !
; 1014          1140  4                [MATCH (DBG$CS_CALL)]:
; 1015          1141  4                    BEGIN
; 1016          1142  4
; 1017          1143  4                    ! Make sure that this is not a SET WATCH command,
; 1018          1144  5                    ! and then set the command kind to call.
; 1019          1145  5                    !
; 1020          1146  5                    IF .CMD_TYPE NEQU EVENT$K_TYPE_WATCH
; 1021          1147  5                    THEN
; 1022          1148  5                        BEGIN
; 1023          1149  5                        CMD_KIND = EVENT$K_KIND_INS;
; 1024          1150  5                        SUB_KIND = EVENT$K_INS_CALL;
; 1025          1151  5                        END
; 1026          1152  6
; 1027          1153  6                    ELSE
; 1028          1154  6                        SYNTAX_ERROR;
; 1029          1155  6
; 1030          1156  6
; 1031          1157  5
; 1032          1158  5
```

```
;  1033          1159  5                END;
;  1034          1160  4
;  1035          1161  4
;  1036          1162  4            ! Parse /BRANCH.
;  1037          1163  4            !
;  1038          1164  4            [MATCH (DBG$CS_BRANCH)]:
;  1039          1165  4                BEGIN
;  1040          1166  5
;  1041          1167  5
;  1042          1168  5                ! Make sure that this is not a SET WATCH command,
;  1043          1169  5                ! and then set the command kind to branch.
;  1044          1170  5                !
;  1045          1171  5                IF .CMD_TYPE NEQU EVENT$K_TYPE_WATCH
;  1046          1172  5                THEN
;  1047          1173  5                    BEGIN
;  1048          1174  6                    CMD_KIND = EVENT$K_KIND_INS;
;  1049          1175  6                    SUB_KIND = EVENT$K_INS_BRAN;
;  1050          1176  6                    END
;  1051          1177  6
;  1052          1178  6                ELSE
;  1053          1179  5                    SYNTAX_ERROR;
;  1054          1180  5
;  1055          1181  5                END;
;  1056          1182  4
;  1057          1183  4
;  1058          1184  4            ! Parse /LINE.
;  1059          1185  4            !
;  1060          1186  4            [MATCH (DBG$CS_LINE)]:
;  1061          1187  4                BEGIN
;  1062          1188  5
;  1063          1189  5
;  1064          1190  5                ! Make sure that this is not a SET WATCH command,
;  1065          1191  5                ! and then set the command kind to line.
;  1066          1192  5                !
;  1067          1193  5                IF .CMD_TYPE NEQU EVENT$K_TYPE_WATCH
;  1068          1194  5                THEN
;  1069          1195  5                    BEGIN
;  1070          1196  6                    CMD_KIND = EVENT$K_KIND_INS;
;  1071          1197  6                    SUB_KIND = EVENT$K_INS_LINE;
;  1072          1198  6                    END
;  1073          1199  6
;  1074          1200  6                ELSE
;  1075          1201  5                    SYNTAX_ERROR;
;  1076          1202  5
;  1077          1203  5                END;
;  1078          1204  4
;  1079          1205  4
;  1080          1206  4            ! Parse /INSTRUCTION
;  1081          1207  4            !           [ = [ <mnemonic> : ( <mnemonic> [ , <mnemonic> ]* ) ]].
;  1082          1208  4            !
;  1083          1209  4            [MATCH (DBG$CS_INSTRUCTION)]:
;  1084          1210  4                BEGIN
;  1085          1211  5
;  1086          1212  5
;  1087          1213  5                ! Make sure that this is not a SET WATCH command,
;  1088          1214  5                ! and then set the command kind to instruction.
;  1089          1215  5
```

```
;  1090          1216  5                !
;  1091          1217  5                IF .CMD_TYPE NEQU EVENT$K_TYPE_WATCH
;  1092          1218  5                THEN
;  1093          1219  6                    BEGIN
;  1094          1220  6                    CMD_KIND = EVENT$K_KIND_INS;
;  1095          1221  6                    SUB_KIND = EVENT$K_INS_EVRY;
;  1096          1222  6
;  1097          1223  6                    ! Look for a possible '='.
;  1098          1224  6                    !
;  1099          1225  6                    IF MATCH (DBG$CS_EQUALS)
;  1100          1226  7                    THEN
;  1101          1227  6                        BEGIN
;  1102          1228  7
;  1103          1229  7                        ! Set the kind to user defined instead of every.
;  1104          1230  7                        !
;  1105          1231  7                        SUB_KIND = EVENT$K_INS_USER;
;  1106          1232  7
;  1107          1233  7                        ! Collect mnemonic(s).
;  1108          1234  7                        !
;  1109          1235  7                        IF MATCH (DBG$CS_L_PAREN)
;  1110          1236  7                        THEN
;  1111          1237  8
;  1112          1238  7                            ! Collect mnemonic(s) and build an noun list.
;  1113          1239  7                            !
;  1114          1240  7                            BEGIN
;  1115          1241  7                            DO
;  1116          1242  7                                BEGIN
;  1117          1243  8
;  1118          1244  8                            ! Get a noun node for a <mnemonic>.
;  1119          1245  9                            !
;  1120          1246  9                            GET_NOUN_NODE;
;  1121          1247  9
;  1122          1248  9                            NOUN_NODE [DBG$L_NOUN_VALUE] =
;  1123          1249  9                                DBG$OPCODE_INDEX(.INPUT_DESC);
;  1124          1250  9
;  1125          1251  9                            END
;  1126          1252  9                        WHILE (MATCH (DBG$CS_COMMA));
;  1127          1253  8
;  1128          1254  8                        ! Match the ')'.
;  1129          1255  8                        !
;  1130          1256  8                        NEED_MATCH (DBG$CS_R_PAREN);
;  1131          1257  8                        END
;  1132          1258  8
;  1133          1259  8                ELSE
;  1134          1260  8                    BEGIN
;  1135          1261  8
;  1136          1262  8                    ! Get a noun node for a <mnemonic>.
;  1137          1263  8                    !
;  1138          1264  7                    GET_NOUN_NODE;
;  1139          1265  8                    NOUN_NODE [DBG$L_NOUN_VALUE] =
;  1140          1266  8                        DBG$OPCODE_INDEX(.INPUT_DESC);
;  1141          1267  8
;  1142          1268  8
;  1143          1269  8
;  1144          1270  8
;  1145          1271  8
;  1146          1272  8
```

```
;  1147          1273  7                        END;
;  1148          1274  7
;  1149          1275  6                    END;
;  1150          1276  6
;  1151          1277  6                END
;  1152          1278  6
;  1153          1279  5            ELSE
;  1154          1280  5                SYNTAX_ERROR;
;  1155          1281  5
;  1156          1282  4            END;
;  1157          1283  4
;  1158          1284  4
;  1159          1285  4            ! Check for ambiguous cases.
;  1160          1286  4            !
;  1161          1287  4            [MATCH(UPLIT BYTE(%ASCIC 'S'))]:
;  1162          1288  4                SIGNAL(DBG$_AMBIGQUAL, 1, UPLIT BYTE(%ASCIC 'S'));
;  1163          1289  4
;  1164          1290  4            [MATCH(UPLIT BYTE(%ASCIC 'NOS'), 3)]:
;  1165          1291  4                SIGNAL(DBG$_AMBIGQUAL, 1, UPLIT BYTE(%ASCIC 'NOS'));
;  1166          1292  4
;  1167          1293  4
;  1168          1294  4            ! Otherwise, we've got an undefined qualifier....
;  1169          1295  4            !
;  1170          1296  4            [OTHERWISE]:
;  1171          1297  4                SYNTAX_ERROR;
;  1172          1298  4
;  1173          1299  4            TES;
;  1174          1300  4
;  1175          1301  3        END;                        ! End of WHILE loop over qualifiers
;  1176          1302  3
;  1177          1303  3    END                             ! End of non-SET STEP command parse
;  1178          1304  3
;  1179          1305  3
;  1180          1306  3    ! Handle the SET STEP command qualifiers.
;  1181          1307  3    !
;  1182          1308  3    ELSE
;  1183          1309  3        BEGIN
;  1184          1310  3
;  1185          1311  3
;  1186          1312  3        ! Parse the SET STEP command according to the following syntax:
;  1187          1313  3        !
;  1188          1314  3        !       [ [NO]SILENT ]
;  1189          1315  3        !       [ [NO]SOURCE ! [NO]SYSTEM ! INTO ! OVER ]
;  1190          1316  3        !       CALLS
;  1191          1317  3        !       RETURN
;  1192          1318  3        !       BRANCH
;  1193          1319  3        !       INSTRUCTION [ = [ <mnemonic> ! [( <mnemonic> [ , <mnemonic> ]* ) ]]
;  1194          1320  3        !
;  1195          1321  3        DO
;  1196          1322  4            BEGIN
;  1197          1323  4            SELECTONE TRUE OF
;  1198          1324  4                SET
;  1199          1325  4
;  1200          1326  4                ! Parse RETURN
;  1201          1327  4                !
;  1202          1328  4                [MATCH (DBG$CS_RETURN)]:
;  1203          1329  4
```

```
;  1204          1330  5                    BEGIN
;  1205          1331  5                    CMD_KIND = EVENT$K_KIND_ACC;
;  1206          1332  5                    SUB_KIND = EVENT$K_ACC_RTRN;
;  1207          1333  4                    END;
;  1208          1334  4
;  1209          1335  4
;  1210          1336  4                ! Parse SOURCE.
;  1211          1337  4                !
;  1212          1338  4                [MATCH (DBG$CS_SOURCE, 2)]:
;  1213          1339  5                    BEGIN
;  1214          1340  5                    SOURCE = TRUE;
;  1215          1341  5                    NOSOURCE = FALSE;
;  1216          1342  4                    END;
;  1217          1343  4
;  1218          1344  4
;  1219          1345  4                ! Parse NOSOURCE.
;  1220          1346  4                !
;  1221          1347  4                [MATCH (DBG$CS_NOSOURCE, 4)]:
;  1222          1348  5                    BEGIN
;  1223          1349  5                    NOSOURCE = TRUE;
;  1224          1350  5                    SOURCE = FALSE;
;  1225          1351  4                    END;
;  1226          1352  4
;  1227          1353  4
;  1228          1354  4                ! Parse SYSTEM.
;  1229          1355  4                !
;  1230          1356  4                [MATCH (DBG$CS_SYSTEM, 2)]:
;  1231          1357  5                    BEGIN
;  1232          1358  5                    SYSTEM = TRUE;
;  1233          1359  5                    NOSYSTEM = FALSE;
;  1234          1360  4                    END;
;  1235          1361  4
;  1236          1362  4
;  1237          1363  4                ! Parse NOSYSTEM.
;  1238          1364  4                !
;  1239          1365  4                [MATCH (DBG$CS_NOSYSTEM, 4)]:
;  1240          1366  5                    BEGIN
;  1241          1367  5                    NOSYSTEM = TRUE;
;  1242          1368  5                    SYSTEM = FALSE;
;  1243          1369  4                    END;
;  1244          1370  4
;  1245          1371  4
;  1246          1372  4                ! Parse SILENT.
;  1247          1373  4                !
;  1248          1374  4                [MATCH (DBG$CS_SILENT, 2)]:
;  1249          1375  5                    BEGIN
;  1250          1376  5                    SILENT = TRUE;
;  1251          1377  5                    NOSILENT = FALSE;
;  1252          1378  4                    END;
;  1253          1379  4
;  1254          1380  4
;  1255          1381  4                ! Parse NOSILENT.
;  1256          1382  4                !
;  1257          1383  4                [MATCH (DBG$CS_NOSILENT, 4)]:
;  1258          1384  5                    BEGIN
;  1259          1385  5                    SILENT = FALSE;
;  1260          1386  5                    NOSILENT = TRUE;
```

```
;  1261          1387  4                END;
;  1262          1388  4
;  1263          1389  4
;  1264          1390  4            ! Parse INTO.
;  1265          1391  4            !
;  1266          1392  4            [MATCH (DBG$CS_INTO)]:
;  1267          1393  5                BEGIN
;  1268          1394  5                INTO = TRUE;
;  1269          1395  5                OVER = FALSE;
;  1270          1396  4                END;
;  1271          1397  4
;  1272          1398  4
;  1273          1399  4            ! Parse OVER.
;  1274          1400  4            !
;  1275          1401  4            [MATCH (DBG$CS_OVER)]:
;  1276          1402  5                BEGIN
;  1277          1403  5                OVER = TRUE;
;  1278          1404  5                INTO = FALSE;
;  1279          1405  4                END;
;  1280          1406  4
;  1281          1407  4
;  1282          1408  4            ! Parse CALLS.
;  1283          1409  4            !
;  1284          1410  4            [MATCH (DBG$CS_CALL)]:
;  1285          1411  5                BEGIN
;  1286          1412  5                CMD_KIND = EVENT$K_KIND_INS;
;  1287          1413  5                SUB_KIND = EVENT$K_INS_CALL;
;  1288          1414  4                END;
;  1289          1415  4
;  1290          1416  4
;  1291          1417  4            ! Parse BRANCH.
;  1292          1418  4            !
;  1293          1419  4            [MATCH (DBG$CS_BRANCH)]:
;  1294          1420  5                BEGIN
;  1295          1421  5                CMD_KIND = EVENT$K_KIND_INS;
;  1296          1422  5                SUB_KIND = EVENT$K_INS_BRAN;
;  1297          1423  4                END;
;  1298          1424  4
;  1299          1425  4
;  1300          1426  4            ! Parse EXCEPTION
;  1301          1427  4            !
;  1302          1428  4            [MATCH (DBG$CS_EXCEPTION)]:
;  1303          1429  5                BEGIN
;  1304          1430  5                CMD_KIND = EVENT$K_KIND_EXC;
;  1305          1431  5                SUB_KIND = EVENT$K_EXC_EXC;
;  1306          1432  4                END;
;  1307          1433  4
;  1308          1434  4
;  1309          1435  4            ! Parse LINE.
;  1310          1436  4            !
;  1311          1437  4            [MATCH (DBG$CS_LINE)]:
;  1312          1438  5                BEGIN
;  1313          1439  5                CMD_KIND = EVENT$K_KIND_INS;
;  1314          1440  5                SUB_KIND = EVENT$K_INS_LINE;
;  1315          1441  4                END;
;  1316          1442  4
;  1317          1443  4
```

```
;  1318          1444  4                ! Parse INSTRUCTION
;  1319          1445  4                !           [ = [ <mnemonic> ! ( <mnemonic> [ , <mnemonic> ]* ) ]].
;  1320          1446  4                !
;  1321          1447  4                [MATCH (DBG$CS_INSTRUCTION)]:
;  1322          1448  5                    BEGIN
;  1323          1449  5                    CMD_KIND = EVENT$K_KIND_INS;
;  1324          1450  5                    SUB_KIND = EVENT$K_INS_EVRY;
;  1325          1451  5
;  1326          1452  5
;  1327          1453  5                    ! Look for a possible '='.
;  1328          1454  5                    !
;  1329          1455  6                    IF MATCH (DBG$CS_EQUALS)
;  1330          1456  5                    THEN
;  1331          1457  6                        BEGIN
;  1332          1458  6
;  1333          1459  6                        ! Set the kind to user defined instead of every.
;  1334          1460  6                        !
;  1335          1461  6                        SUB_KIND = EVENT$K_INS_USER;
;  1336          1462  6
;  1337          1463  6                        ! Collect mnemonic(s).
;  1338          1464  6                        !
;  1339          1465  7                        IF MATCH (DBG$CS_L_PAREN)
;  1340          1466  6                        THEN
;  1341          1467  6
;  1342          1468  6                            ! Collect mnemonic(s) and build an noun list.
;  1343          1469  6                            !
;  1344          1470  7                            BEGIN
;  1345          1471  7                            DO
;  1346          1472  8                                BEGIN
;  1347          1473  8
;  1348          1474  8                                ! Get a noun node for a <mnemonic>.
;  1349          1475  8                                !
;  1350          1476  8                                GET_NOUN_NODE;
;  1351          1477  8
;  1352          1478  8                                NOUN_NODE [DBG$L_NOUN_VALUE] =
;  1353          1479  8                                    DBG$OPCODE_INDEX(.INPUT_DESC);
;  1354          1480  8
;  1355          1481  8                                END
;  1356          1482  7                            WHILE (MATCH (DBG$CS_COMMA));
;  1357          1483  7
;  1358          1484  7
;  1359          1485  7                            ! Match the ')'.
;  1360          1486  7                            !
;  1361          1487  7                            NEED_MATCH (DBG$CS_R_PAREN);
;  1362          1488  7                            END
;  1363          1489  7
;  1364          1490  6                    ELSE
;  1365          1491  7                        BEGIN
;  1366          1492  7
;  1367          1493  7                        ! Get a noun node for a <mnemonic>.
;  1368          1494  7                        !
;  1369          1495  7                        GET_NOUN_NODE;
;  1370          1496  7                        NOUN_NODE [DBG$L_NOUN_VALUE] =
;  1371          1497  7                            DBG$OPCODE_INDEX(.INPUT_DESC);
;  1372          1498  6                        END;
;  1373          1499  6
;  1374          1500  5                    END;
```

```
;  1375          1501  5
;  1376          1502  4                    END;
;  1377          1503  4
;  1378          1504  4
;  1379          1505  4                    ! Check for ambiguous cases.
;  1380          1506  4                    !
;  1381          1507  4                    [MATCH(UPLIT BYTE(%ASCIC 'S'))]:
;  1382          1508  4                        SIGNAL(DBG$_AMBIGQUAL, 1, UPLIT BYTE(%ASCIC 'S'));
;  1383          1509  4
;  1384          1510  4                    [MATCH(UPLIT BYTE(%ASCIC 'NOS'), 3)]:
;  1385          1511  4                        SIGNAL(DBG$_AMBIGQUAL, 1, UPLIT BYTE(%ASCIC 'NOS'));
;  1386          1512  4
;  1387          1513  4
;  1388          1514  4                    ! Otherwise, we've got an undefined qualifier (or something).
;  1389          1515  4                    !
;  1390          1516  4                    [OTHERWISE]:
;  1391          1517  4                        SYNTAX_ERROR;
;  1392          1518  4
;  1393          1519  4                    TES;
;  1394          1520  4
;  1395          1521  4                END
;  1396          1522  4
;  1397          1523  3            WHILE (MATCH (DBG$CS_COMMA));
;  1398          1524  3
;  1399          1525  2            END;                            ! End of SET STEP qualifier parsing
;  1400          1526  2
;  1401          1527  2
;  1402          1528  2        ! Build up the adverb list based on what we've got so far.
;  1403          1529  2        ! Note that if we've reached this point, we should have
;  1404          1530  2        ! all qualifiers done, and the working variables reflect
;  1405          1531  2        ! the command desired.
;  1406          1532  2        !
;  1407          1533  2        ! Set up the command kind and possibly the sub-kind adverbs.
;  1408          1534  2        !
;  1409          1535  2        GET_ADVERB_NODE;
;  1410          1536  2        SELECTONE .CMD_KIND OF
;  1411          1537  2            SET
;  1412          1538  2            [EVENT$K_KIND_ACC]:
;  1413          1539  2                ADVERB_NODE [DBG$B_ADVERB_LITERAL] =
;  1414          1540  2                (SELECTONE .SUB_KIND OF
;  1415          1541  2                    SET
;  1416          1542  2                    [EVENT$K_ACC_READ]:
;  1417          1543  2                        ADVERB_READ;
;  1418          1544  2                    [EVENT$K_ACC_WRIT]:
;  1419          1545  2                        ADVERB_WRITE;
;  1420          1546  2                    [EVENT$K_ACC_MDFY]:
;  1421          1547  2                        ADVERB_MODIFY;
;  1422          1548  2                    [EVENT$K_ACC_EXEC]:
;  1423          1549  2                        ADVERB_EXECUTE;
;  1424          1550  2                    [EVENT$K_ACC_RTRN]:
;  1425          1551  2                        ADVERB_RETURN;
;  1426          1552  2                    TES
;  1427          1553  2                );
;  1428          1554  2
;  1429          1555  2
;  1430          1556  2            [EVENT$K_KIND_EXC]:
;  1431          1557  2                ADVERB_NODE [DBG$B_ADVERB_LITERAL] = ADVERB_EXCEPTION;
```

```
;  1432     1558  2
;  1433     1559  2
;  1434     1560  2              [EVENT$K_KIND_INS]:
;  1435     1561  2                  ADVERB_NODE [DBG$B_ADVERB_LITERAL] =
;  1436     1562  2                  (SELECTONE .SUB_KIND OF
;  1437     1563  3                      SET
;  1438     1564  3                      [EVENT$K_INS_CALL]:
;  1439     1565  3                          ADVERB_CALL;
;  1440     1566  3                      [EVENT$K_INS_BRAN]:
;  1441     1567  3                          ADVERB_BRANCH;
;  1442     1568  3                      [EVENT$K_INS_EVRY]:
;  1443     1569  3                          ADVERB_INSTRUCTION;
;  1444     1570  3                      [EVENT$K_INS_USER]:
;  1445     1571  3                          ADVERB_OPCODE;
;  1446     1572  3                      [EVENT$K_INS_LINE]:
;  1447     1573  3                          ADVERB_LINE;
;  1448     1574  3                      TES
;  1449     1575  2                  );
;  1450     1576  2
;  1451     1577  2              TES;
;  1452     1578  2
;  1453     1579  2
;  1454     1580  2          ! If the /AFTER, /UNTIL, and /EVERY are not at their defaults,
;  1455     1581  2          ! setup an adverb node for the problem boys.
;  1456     1582  2          !
;  1457     1583  2          IF .AFTER_COUNT NEQU 1
;  1458     1584  2          THEN
;  1459     1585  3              BEGIN
;  1460     1586  3              GET_ADVERB_NODE;
;  1461     1587  3
;  1462     1588  3
;  1463     1589  3              ! If the /AFTER count is 0, then this is really a kludge ah, I
;  1464     1590  3              ! mean /TEMPORARY.
;  1465     1591  3              !
;  1466     1592  3              IF .AFTER_COUNT EQLU 0
;  1467     1593  3              THEN
;  1468     1594  3                  ADVERB_NODE [DBG$B_ADVERB_LITERAL] = ADVERB_TEMPORARY
;  1469     1595  3              ELSE
;  1470     1596  4                  BEGIN
;  1471     1597  4                  ADVERB_NODE [DBG$B_ADVERB_LITERAL] = ADVERB_AFTER;
;  1472     1598  4                  ADVERB_NODE [DBG$L_ADVERB_VALUE] = .AFTER_COUNT;
;  1473     1599  3                  END;
;  1474     1600  2              END;
;  1475     1601  2
;  1476     1602  2
;  1477     1603  2          ! If the /SILENT or /NOSILENT qualifier is present, make an Adverb Node
;  1478     1604  2          ! for whichever one was specified last.
;  1479     1605  2          !
;  1480     1606  2          IF .SILENT
;  1481     1607  2          THEN
;  1482     1608  3              BEGIN
;  1483     1609  3              GET_ADVERB_NODE;
;  1484     1610  3              ADVERB_NODE [DBG$B_ADVERB_LITERAL] = ADVERB_SILENT;
;  1485     1611  3              END
;  1486     1612  3
;  1487     1613  2          ELSE IF .NOSILENT
;  1488     1614  2          THEN
```

```
;  1489          1615  3          BEGIN
;  1490          1616  3          GET_ADVERB_NODE;
;  1491          1617  3          ADVERB_NODE [DBG$B_ADVERB_LITERAL] = ADVERB_NOSILENT;
;  1492          1618  3          END;
;  1493          1619
;  1494          1620
;  1495          1621  2      ! If the /TEMPORARY qualifier is present, make an Adverb Node for it.
;  1496          1622  2      !
;  1497          1623  2      IF .TEMPORARY
;  1498          1624  2      THEN
;  1499          1625  3          BEGIN
;  1500          1626  3          GET_ADVERB_NODE;
;  1501          1627  3          ADVERB_NODE [DBG$B_ADVERB_LITERAL] = ADVERB_TEMPORARY;
;  1502          1628  3          END;
;  1503          1629
;  1504          1630
;  1505          1631  2      ! If the /SOURCE or /NOSOURCE qualifier was present, make an Adverb Node
;  1506          1632  2      ! for it.
;  1507          1633  2      !
;  1508          1634  2      IF .SOURCE
;  1509          1635  2      THEN
;  1510          1636  3          BEGIN
;  1511          1637  3          GET_ADVERB_NODE;
;  1512          1638  3          ADVERB_NODE [DBG$B_ADVERB_LITERAL] = ADVERB_SOURCE;
;  1513          1639  3          END;
;  1514          1640
;  1515          1641  2      IF .NOSOURCE
;  1516          1642  2      THEN
;  1517          1643  3          BEGIN
;  1518          1644  3          GET_ADVERB_NODE;
;  1519          1645  3          ADVERB_NODE [DBG$B_ADVERB_LITERAL] = ADVERB_NOSOURCE;
;  1520          1646  3          END;
;  1521          1647  2
;  1522          1648  2
;  1523          1649  2      ! Handle /[NO]SYSTEM's, /INTO's or /OVER's.
;  1524          1650  2      !
;  1525          1651  2      IF .CMD_KIND EQLU EVENT$K_KIND_INS OR
;  1526          1652  2         (.CMD_KIND EQLU EVENT$K_KIND_ACC AND
;  1527          1653  2          .SUB_KIND EQLU EVENT$K_ACC_RTRN)
;  1528          1654  2      THEN
;  1529          1655  3          BEGIN
;  1530          1656  3          IF .SYSTEM
;  1531          1657  3          THEN
;  1532          1658  4              BEGIN
;  1533          1659  4              GET_ADVERB_NODE;
;  1534          1660  4              ADVERB_NODE [DBG$B_ADVERB_LITERAL] = ADVERB_SYSTEM;
;  1535          1661  4              END;
;  1536          1662  3
;  1537          1663  3          IF .NOSYSTEM
;  1538          1664  3          THEN
;  1539          1665  4              BEGIN
;  1540          1666  4              GET_ADVERB_NODE;
;  1541          1667  4              ADVERB_NODE [DBG$B_ADVERB_LITERAL] = ADVERB_NOSYSTEM;
;  1542          1668  4              END;
;  1543          1669  3
;  1544          1670  3          IF .INTO
;  1545          1671  3          THEN
```

```
;  1546          1672  4              BEGIN
;  1547          1673  4              GET_ADVERB_NODE;
;  1548          1674  4              ADVERB_NODE [DBG$B_ADVERB_LITERAL] = ADVERB_INTO;
;  1549          1675  3              END;
;  1550          1676  3
;  1551          1677  3          IF .OVER
;  1552          1678  3          THEN
;  1553          1679  4              BEGIN
;  1554          1680  4              GET_ADVERB_NODE;
;  1555          1681  4              ADVERB_NODE [DBG$B_ADVERB_LITERAL] = ADVERB_OVER;
;  1556          1682  3              END;
;  1557          1683  3
;  1558          1684  2          END;
;  1559          1685  2
;  1560          1686  2
;  1561          1687  2      ! If the command kind is an access kind, get the <address> [ , <address> ]*
;  1562          1688  2      ! as long as an <address> is terminated with a ','.
;  1563          1689  2      !
;  1564          1690  2      IF .CMD_TYPE NEQU EVENT$K_TYPE_STEPS AND
;  1565          1691  2         .CMD_KIND EQLU EVENT$K_KIND_ACC
;  1566          1692  2      THEN
;  1567          1693  3          BEGIN
;  1568          1694  3          DO
;  1569          1695  4              BEGIN
;  1570          1696  4
;  1571          1697  4
;  1572          1698  4              ! Get a noun node for an <address>.
;  1573          1699  4              !
;  1574          1700  4              GET_NOUN_NODE;
;  1575          1701  4
;  1576          1702  4
;  1577          1703  4              ! Get an <address> - the is a status warning
;  1578          1704  4              ! if more is on the line. Surrounding this call, we
;  1579          1705  4              ! turn on a flag saying we are picking up the address in
;  1580          1706  4              ! a set break command. This is used in DBGPARSER to
;  1581          1707  4              ! resolve potential ambiguities in SET BRE . DO (command)
;  1582          1708  4              !
;  1583          1709  4              DBG$GB_SET_BREAK_FLAG = TRUE;
;  1584          1710  4              STATUS = DBG$NPARSE_ADDRESS(.INPUT_DESC,
;  1585          1711  4                           NOUN_NODE [DBG$L_NOUN_VALUE],
;  1586          1712  4                           .DBG$GB_RADIX[DBG$B_RADIX_INPUT],
;  1587          1713  4                           TOKEN$K_TERM_CMWHDO, .MESSAGE_VECT);
;  1588          1714  4              DBG$GB_SET_BREAK_FLAG = FALSE;
;  1589          1715  4
;  1590          1716  4
;  1591          1717  4              ! If the status indicates success, there is no
;  1592          1718  4              ! more on the line, and we exit this loop.
;  1593          1719  4              !
;  1594          1720  4              IF .STATUS EQLU STS$K_SUCCESS
;  1595          1721  4              THEN
;  1596          1722  4                  EXITLOOP;
;  1597          1723  4
;  1598          1724  4
;  1599          1725  4              ! If the status is not a warning, something is
;  1600          1726  4              ! wrong, so return with a failure.
;  1601          1727  4              !
;  1602          1728  4              IF .STATUS NEQU STS$K_WARNING
```

```
;  1603     1729  4              THEN
;  1604     1730  4                  RETURN .STATUS
;  1605     1731  4
;  1606     1732  4              END
;  1607     1733  3          WHILE (MATCH (DBG$CS_COMMA));
;  1608     1734  3
;  1609     1735  2          END;
;  1610     1736  2
;  1611     1737  2
;  1612     1738  2          ! Check the input for a <cr> after parsing <address> [ , <address> ]*
;  1613     1739  2          ! or <mnemonic> [ , <mnemonic> ].  If there's nothing left on the
;  1614     1740  2          ! line we can leave.
;  1615     1741  2          !
;  1616     1742  2          IF MATCH (DBG$CS_CR)
;  1617     1743  2          THEN RETURN STS$K_SUCCESS;
;  1618     1744  2
;  1619     1745  2
;  1620     1746  2          ! We didn't match a <cr>, there's more on the line.  If this is a STEP
;  1621     1747  2          ! command, get the step count....
;  1622     1748  2          !
;  1623     1749  2          IF .CMD_TYPE EQLU EVENT$K_TYPE_STEPS
;  1624     1750  2          THEN
;  1625     1751  2              IF .VERB_NODE [DBG$B_VERB_COMPOSITE] EQLU EVENT$K_STEP
;  1626     1752  2              THEN
;  1627     1753  3                  BEGIN
;  1628     1754  3
;  1629     1755  3
;  1630     1756  3                  ! Get an adverb node for a <step count>.
;  1631     1757  3                  !
;  1632     1758  3                  GET_ADVERB_NODE;
;  1633     1759  3
;  1634     1760  3
;  1635     1761  3                  ! Get a <step count> - there is a status warning
;  1636     1762  3                  ! if more is on the line.
;  1637     1763  3                  !
;  1638     1764  3                  ! Fill in the details.
;  1639     1765  3                  !
;  1640     1766  3                  ADVERB_NODE [DBG$B_ADVERB_LITERAL] = ADVERB_AFTER;
;  1641     1767  3                  IF NOT DBG$NSAVE_DECIMAL_INTEGER
;  1642     1768  3                         (.INPUT_DESC,
;  1643     1769  3                          ADVERB_NODE [DBG$L_ADVERB_VALUE],
;  1644     1770  3                          .MESSAGE_VECT
;  1645     1771  3                         )
;  1646     1772  3                  THEN
;  1647     1773  3                      RETURN STS$K_SEVERE
;  1648     1774  3                  ELSE
;  1649     1775  3                      RETURN STS$K_SUCCESS;
;  1650     1776  3                  END
;  1651     1777  2              ELSE
;  1652     1778  2                  SYNTAX_ERROR;
;  1653     1779  2
;  1654     1780  2
;  1655     1781  2          ! This isn't a STEP command, but there's more on the line - maybe
;  1656     1782  2          ! a WHEN or a DO....
;  1657     1783  2          !
;  1658     1784  2          ! Parse a possible WHEN (<expression>).
;  1659     1785  2          !
```

```
;  1660          1786  3      IF MATCH (DBG$CS_WHEN)
;  1661          1787  3      THEN
;  1662          1788  3          BEGIN
;  1663          1789  3
;  1664          1790  3
;  1665          1791  3          ! Match the '('.
;  1666          1792  3          !
;  1667          1793  3          NEED_MATCH (DBG$CS_L_PAREN);
;  1668          1794  3
;  1669          1795  3
;  1670          1796  3          ! Get a new adverb node for the conditional expression.
;  1671          1797  3          !
;  1672          1798  3          GET_ADVERB_NODE;
;  1673          1799  3
;  1674          1800  3
;  1675          1801  3          ! Flag a 'WHEN' and save the <expression>.
;  1676          1802  3          !
;  1677          1803  3          ADVERB_NODE [DBG$B_ADVERB_LITERAL] = ADVERB_WHEN;
;  1678          1804  3          DBG$NSAVE_BREAK_BUFFER (.INPUT_DESC, ADVERB_NODE [DBG$L_ADVERB_VALUE]);
;  1679          1805  3          END;
;  1680          1806  3
;  1681          1807  3
;  1682          1808  3      ! Parse a Possible DO <command list>.
;  1683          1809  3      !
;  1684          1810  3      IF MATCH (DBG$CS_DO)
;  1685          1811  3      THEN
;  1686          1812  3          BEGIN
;  1687          1813  3
;  1688          1814  3
;  1689          1815  3          ! Match the '('.
;  1690          1816  3          !
;  1691          1817  3          NEED_MATCH (DBG$CS_L_PAREN);
;  1692          1818  3
;  1693          1819  3
;  1694          1820  3          ! Get a new adverb node for the command list.
;  1695          1821  3          !
;  1696          1822  3          GET_ADVERB_NODE;
;  1697          1823  3
;  1698          1824  3
;  1699          1825  3          ! Flag a 'DO' and save the <command list>.
;  1700          1826  3          !
;  1701          1827  3          ADVERB_NODE [DBG$B_ADVERB_LITERAL] = ADVERB_DO;
;  1702          1828  3          DBG$NSAVE_BREAK_BUFFER (.INPUT_DESC, ADVERB_NODE [DBG$L_ADVERB_VALUE]);
;  1703          1829  3          RETURN STS$K_SUCCESS;
;  1704          1830  3          END;
;  1705          1831  2
;  1706          1832  2
;  1707          1833  2      ! If there's more on the line, it's nothing we recognise.  Signal
;  1708          1834  2      ! a syntax error.
;  1709          1835  2      !
;  1710          1836  2      IF MATCH (DBG$CS_CR) THEN RETURN STS$K_SUCCESS;
;  1711          1837  2      .MESSAGE_VECT = DBG$NSYNTAX_ERROR (DBG$NNEXT_WORD (.INPUT_DESC));
;  1712          1838  2      RETURN STS$K_SEVERE;
;  1713          1839  1      END;
```

```
                                                        .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                                    52 45 54 46 41 05  00000 P.AAA:  .ASCII  <5>\AFTER\
                              59 52 41 52 4F 50 4D 45 54 09  00006 P.AAB:  .ASCII  <9>\TEMPORARY\
                                       54 4E 45 4C 49 53 06  00010 P.AAC:  .ASCII  <6>\SILENT\
                                 54 4E 45 4C 49 53 4F 4E 08  00017 P.AAD:  .ASCII  <8>\NOSILENT\
                                             4E 45 48 57 04  00020 P.AAE:  .ASCII  <4>\WHEN\
                                                   4F 44 02  00025 P.AAF:  .ASCII  <2>\DO\
                                          44 41 45 52 04  00028 P.AAG:  .ASCII  <4>\READ\
                                       45 54 49 52 57 05  0002D P.AAH:  .ASCII  <5>\WRITE\
                                    59 46 49 44 4F 4D 06  00033 P.AAI:  .ASCII  <6>\MODIFY\
                                 45 54 55 43 45 58 45 07  0003A P.AAJ:  .ASCII  <7>\EXECUTE\
                                    4E 52 55 54 45 52 06  00042 P.AAK:  .ASCII  <6>\RETURN\
                                    45 43 52 55 4F 53 06  00049 P.AAL:  .ASCII  <6>\SOURCE\
                                 45 43 52 55 4F 53 4F 4E 08  00050 P.AAM:  .ASCII  <8>\NOSOURCE\
                                    4D 45 54 53 59 53 06  00059 P.AAN:  .ASCII  <6>\SYSTEM\
                                 4D 45 54 53 59 53 4F 4E 08  00060 P.AAO:  .ASCII  <8>\NOSYSTEM\
                                          4F 54 4E 49 04  00069 P.AAP:  .ASCII  <4>\INTO\
                                          52 45 56 4F 04  0006E P.AAQ:  .ASCII  <4>\OVER\
                              4E 4F 49 54 50 45 43 58 45 09  00073 P.AAR:  .ASCII  <9>\EXCEPTION\
                                       53 4C 4C 41 43 05  0007D P.AAS:  .ASCII  <5>\CALLS\
                                    48 43 4E 41 52 42 06  00083 P.AAT:  .ASCII  <6>\BRANCH\
                                          45 4E 49 4C 04  0008A P.AAU:  .ASCII  <4>\LINE\
                  4E 4F 49 54 43 55 52 54 53 4E 49 0B  0008F P.AAV:  .ASCII  <11>\INSTRUCTION\
                                                3D 01  0009B P.AAW:  .ASCII  <1>\=\
                                                28 01  0009D P.AAX:  .ASCII  <1>\(\
                                                29 01  0009F P.AAY:  .ASCII  <1>\)\
                                                2C 01  000A1 P.AAZ:  .ASCII  <1>\,\
                                                2F 01  000A3 P.ABA:  .ASCII  <1>\/\
                                                3A 01  000A5 P.ABB:  .ASCII  <1>\:\
                                                0D 01  000A7 P.ABC:  .BYTE   1, 13
                                                53 01  000A9 P.ABD:  .ASCII  <1>\S\
                                                53 01  000AB P.ABE:  .ASCII  <1>\S\
                                          53 4F 4E 03  000AD P.ABF:  .ASCII  <3>\NOS\
                                          53 4F 4E 03  000B1 P.ABG:  .ASCII  <3>\NOS\
                                                53 01  000B5 P.ABH:  .ASCII  <1>\S\
                                                53 01  000B7 P.ABI:  .ASCII  <1>\S\
                                          53 4F 4E 03  000B9 P.ABJ:  .ASCII  <3>\NOS\
                                          53 4F 4E 03  000BD P.ABK:  .ASCII  <3>\NOS\

                                                        DBG$CS_AFTER=          P.AAA
                                                        DBG$CS_TEMPORARY=      P.AAB
                                                        DBG$CS_SILENT=         P.AAC
                                                        DBG$CS_NOSILENT=       P.AAD
                                                        DBG$CS_WHEN=           P.AAE
                                                        DBG$CS_DO=             P.AAF
                                                        DBG$CS_READ=           P.AAG
                                                        DBG$CS_WRITE=          P.AAH
                                                        DBG$CS_MODIFY=         P.AAI
                                                        DBG$CS_EXECUTE=        P.AAJ
                                                        DBG$CS_RETURN=         P.AAK
                                                        DBG$CS_SOURCE=         P.AAL
                                                        DBG$CS_NOSOURCE=       P.AAM
                                                        DBG$CS_SYSTEM=         P.AAN
                                                        DBG$CS_NOSYSTEM=       P.AAO
                                                        DBG$CS_INTO=           P.AAP
                                                        DBG$CS_OVER=           P.AAQ
```

```
                                                     1  8
                                DBG$CS_EXCEPTION=   P.AAR
                                DBG$CS_CALL=        P.AAS
                                DBG$CS_BRANCH=      P.AAT
                                DBG$CS_LINE=        P.AAU
                                DBG$CS_INSTRUCTION= P.AAV
                                DBG$CS_EQUALS=      P.AAW
                                DBG$CS_L_PAREN=     P.AAX
                                DBG$CS_R_PAREN=     P.AAY
                                DBG$CS_COMMA=       P.AAZ
                                DBG$CS_SLASH=       P.ABA
                                DBG$CS_COLON=       P.ABB
                                DBG$CS_CR=          P.ABC

                                                .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                          0FFC 00000            .ENTRY  DBG$EVENT_SYNTAX, Save R2,R3,R4,R5,R6,R7,-  ; 0589
                                                        R8,R9,R10,R11
                       5E    24  C2 00002             SUBL2   #36, SP                             ;
                       50 08 AC  D0 00005             MOVL    VERB_NODE, R0                       ; 0715
                          04 A0  7C 00009             CLRQ    4(R0)                               ;
                       58 04 A0  9E 0000C             MOVAB   4(R0), ADVERB_LINK                  ; 0721
                       59 08 A0  9E 00010             MOVAB   8(R0), NOUN_LINK                    ; 0722
                       50 01 A0  9A 00014             MOVZBL  1(R0), R0                           ; 0727
                       01        50  91 00018         CMPB    R0, #1                              ; 0731
                                 04  12 0001B         BNEQ    1$                                  ;
                                 56  D4 0001D         CLRL    CMD_TYPE                            ; 0733
                                 08  11 0001F         BRB     2$                                  ; 0734
                       0E        50  91 00021 1$:     CMPB    R0, #14                             ; 0739
                                 0A  12 00024         BNEQ    3$                                  ;
                       56        01  D0 00026         MOVL    #1, CMD_TYPE                        ; 0741
                                 57  D4 00029 2$:     CLRL    CMD_KIND                            ; 0742
                       55        03  D0 0002B         MOVL    #3, SUB_KIND                        ; 0743
                                 3A  11 0002E         BRB     7$                                  ; 0727
                       11        50  91 00030 3$:     CMPB    R0, #17                             ; 0747
                                 08  12 00033         BNEQ    4$                                  ;
                       56        02  7D 00035         MOVQ    #2, CMD_TYPE                        ; 0749
                       55        02  D0 00038         MOVL    #2, SUB_KIND                        ; 0751
                                 2D  11 0003B         BRB     7$                                  ; 0727
                       0B        50  91 0003D 4$:     CMPB    R0, #11                             ; 0755
                                 28  1F 00040         BLSSU   7$                                  ;
                       0C        50  91 00042         CMPB    R0, #12                             ;
                                 23  1A 00045         BGTRU   7$                                  ;
                       56        03  D0 00047         MOVL    #3, CMD_TYPE                        ; 0757
                       51 00000000G  00  D0 0004A     MOVL    DBG$GB_STP_PTR, R1                  ; 0758
                       55        61  9A 00051         MOVZBL  (R1), SUB_KIND                      ;
                       04        55  D1 00054         CMPL    SUB_KIND, #4                        ; 0759
                                 04  12 00057         BNEQ    5$                                  ;
                                 57  D4 00059         CLRL    CMD_KIND                            ; 0761
                                 0D  11 0005B         BRB     7$                                  ;
                       0F        55  D1 0005D 5$:     CMPL    SUB_KIND, #15                       ; 0763
                                 05  12 00060         BNEQ    6$                                  ;
                       57        02  D0 00062         MOVL    #2, CMD_KIND                        ; 0765
                                 03  11 00065         BRB     7$                                  ;
                       57        01  D0 00067 6$:     MOVL    #1, CMD_KIND                        ; 0768
                    20 AE        01  D0 0006A 7$:     MOVL    #1, AFTER_COUNT                     ; 0777
                                 5A  7C 0006E         CLRQ    NOSILENT                            ; 0780
```

```
                        14   AE  7C 00070          CLRQ    SOURCE                                          ; 0781
                        0C   AE  7C 00073          CLRQ    SYSTEM                                          ; 0783
                        04   AE  7C 00076          CLRQ    INTO                                            ; 0785
                             6E  D4 00079          CLRL    OVER                                            ; 0786
                        1C   AE  D4 0007B          CLRL    28(SP)                                          ; 0787
                   0C        50  91 0007E          CMPB    R0, #12                                         ;
                             12  12 00081          BNEQ    8$                                              ;
                        1C   AE  D6 00083          INCL    28(SP)                                          ;
                   51 00000000G  00  D0 00086          MOVL    DBG$GB_STP_PTR, R1                          ; 0789
5B            61   01           0C  EF 00089          EXTZV   #12, #7, (R1), SILENT                       ;
                   5B           5B  D2 00092          MCOML   SILENT, SILENT                              ;
                   53        04 AC  D0 00095 8$:      MOVL    INPUT_DESC, R3                              ; 0818
                   08           50  91 00099          CMPB    R0, #T1                                     ; 0796
                                03  12 0009C          BNEQ    9$                                          ;
                                    0404 31 0009E      BRW     72$                                         ;
                   5A               01  D0 000A1 9$:      MOVL    #1, NOSILENT                            ; 0817
                                    01  DD 000A4 10$:     PUSHL   #1                                      ; 0818
                        00000000'   EF  9F 000A6          PUSHAB  DBG$CS_SLASH                            ;
                                    53  DD 000AC          PUSHL   R3                                      ;
              00000000G             00  03 FB 000AE      CALLS   #3, DBG$NMATCH                          ;
                                    03  50 E8 000B5      BLBS    R0, 11$                                 ;
                                    0682 31 000B8      BRW     103$                                        ;
                                    01  DD 000BB 11$:     PUSHL   #1                                      ; 0826
                        00000000'   EF  9F 000BD          PUSHAB  DBG$CS_AFTER                            ;
                                    53  DD 000C3          PUSHL   R3                                      ;
              00000000G             00  03 FB 000C5      CALLS   #3, DBG$NMATCH                          ;
                                    01  50 D1 000CC      CMPL    R0, #1                                  ;
                                    51  12 000CF      BNEQ    15$                                         ;
                                    56  D5 000D1      TSTL    CMD_TYPE                                    ; 0828
                                    0A  13 000D3      BEQL    12$                                         ;
                   01               56  D1 000D5      CMPL    CMD_TYPE, #1                                ; 0829
                                    05  13 000D8      BEQL    12$                                         ;
                   02               56  D1 000DA      CMPL    CMD_TYPE, #2                                ; 0830
                                    65  12 000DD      BNEQ    16$                                         ;
                                    01  DD 000DF 12$:     PUSHL   #1                                      ; 0837
                        00000000'   EF  9F 000E1          PUSHAB  DBG$CS_COLON                            ;
                                    53  DD 000E7          PUSHL   R3                                      ;
              00000000G             00  03 FB 000E9      CALLS   #3, DBG$NMATCH                          ;
                                    1A  50 E8 000F0      BLBS    R0, 14$                                 ;
                                    01  DD 000F3          PUSHL   #1                                      ;
                        00000000'   EF  9F 000F5          PUSHAB  DBG$CS_CR                               ;
                                    53  DD 000FB          PUSHL   R3                                      ;
              00000000G             00  03 FB 000FD      CALLS   #3, DBG$NMATCH                          ;
                                    03  50 E8 00104      BLBS    R0, 13$                                 ;
                                    0990 31 00107      BRW     143$                                        ;
                                    093F 31 0010A 13$:     BRW     139$                                    ;
                             0C     AC  DD 0010D 14$:     PUSHL   MESSAGE_VECT                            ; 0845
                             24     AE  9F 00110          PUSHAB  AFTER_COUNT                             ; 0843
                                    53  DD 00113          PUSHL   R3                                      ;
              00000000G             00  03 FB 00115      CALLS   #3, DBG$NSAVE_DECIMAL_INTEGER           ;
                                    85  50 E8 0011C      BLBS    R0, 10$                                 ;
                                    098E 31 0011F      BRW     145$                                        ; 0848
                                    01  DD 00122 15$:     PUSHL   #1                                      ; 0860
                        00000000'   EF  9F 00124          PUSHAB  DBG$CS_TEMPORARY                        ;
                                    53  DD 0012A          PUSHL   R3                                      ;
              00000000G             00  03 FB 0012C      CALLS   #3, DBG$NMATCH                          ;
                                    01  50 D1 00133      CMPL    R0, #1                                  ;
```

```
                                    14  12 00136          BNEQ    18$                                                          ;
                                    56  D5 00138          TSTL    CMD_TYPE                                                     ; 0864
                                    0A  13 0013A          BEQL    17$                                                          ;
                              01    56  D1 0013C          CMPL    CMD_TYPE, #1                                                 ; 0865
                                    05  13 0013F          BEQL    17$                                                          ;
                              02    56  D1 00141          CMPL    CMD_TYPE, #2                                                 ; 0866
                                    5B  12 00144 16$:     BNEQ    21$                                                          ;
                           18 AE    01  D0 00146 17$:     MOVL    #1, TEMPORARY                                                ; 0873
                                    5B  11 0014A          BRB     23$                                                          ; 0864
                                    02  DD 0014C 18$:     PUSHL   #2                                                           ; 0884
                    00000000'       EF  9F 0014E          PUSHAB  DBG$CS_SILENT                                                ;
                                    53  DD 00154          PUSHL   R3                                                           ;
                00000000G   00      03  FB 00156          CALLS   #3, DBG$NMATCH                                               ;
                            01      50  D1 0015D          CMPL    R0, #1                                                       ;
                                    07  12 00160          BNEQ    19$                                                          ;
                              5B    01  D0 00162          MOVL    #1, SILENT                                                   ; 0886
                                    5A  D4 00165          CLRL    NOSILENT                                                     ; 0887
                                    64  11 00167          BRB     26$                                                          ; 0820
                                    04  DD 00169 19$:     PUSHL   #4                                                           ; 0893
                    00000000'       EF  9F 0016B          PUSHAB  DBG$CS_NOSILENT                                              ;
                                    53  DD 00171          PUSHL   R3                                                           ;
                00000000G   00      03  FB 00173          CALLS   #3, DBG$NMATCH                                               ;
                            01      50  D1 0017A          CMPL    R0, #1                                                       ;
                                    05  12 0017D          BNEQ    20$                                                          ;
                              5A    01  7D 0017F          MOVQ    #1, NOSILENT                                                 ; 0895
                                    6F  11 00182          BRB     29$                                                          ; 0820
                                    01  DD 00184 20$:     PUSHL   #1                                                           ; 0902
                    00000000'       EF  9F 00186          PUSHAB  DBG$CS_READ                                                  ;
                                    53  DD 0018C          PUSHL   R3                                                           ;
                00000000G   00      03  FB 0018E          CALLS   #3, DBG$NMATCH                                               ;
                            01      50  D1 00195          CMPL    R0, #1                                                       ;
                                    0F  12 00198          BNEQ    24$                                                          ;
                                    56  D5 0019A          TSTL    CMD_TYPE                                                     ; 0909
                                    05  13 0019C          BEQL    22$                                                          ;
                              01    56  D1 0019E          CMPL    CMD_TYPE, #1                                                 ; 0910
                                    6F  12 001A1 21$:     BNEQ    31$                                                          ;
                                    57  D4 001A3 22$:     CLRL    CMD_KIND                                                     ; 0913
                                    55  D4 001A5          CLRL    SUB_KIND                                                     ; 0914
                                    70  11 001A7 23$:     BRB     33$                                                          ; 0909
                                    01  DD 001A9 24$:     PUSHL   #1                                                           ; 0924
                    00000000'       EF  9F 001AB          PUSHAB  DBG$CS_WRITE                                                 ;
                                    53  DD 001B1          PUSHL   R3                                                           ;
                00000000G   00      03  FB 001B3          CALLS   #3, DBG$NMATCH                                               ;
                            01      50  D1 001BA          CMPL    R0, #1                                                       ;
                                    10  12 001BD          BNEQ    27$                                                          ;
                                    56  D5 001BF          TSTL    CMD_TYPE                                                     ; 0931
                                    05  13 001C1          BEQL    25$                                                          ;
                              01    56  D1 001C3          CMPL    CMD_TYPE, #1                                                 ; 0932
                                    6E  12 001C6          BNEQ    35$                                                          ;
                                    57  D4 001C8 25$:     CLRL    CMD_KIND                                                     ; 0935
                              55    01  D0 001CA          MOVL    #1, SUB_KIND                                                 ; 0936
                                    6F  11 001CD 26$:     BRB     37$                                                          ; 0931
                                    01  DD 001CF 27$:     PUSHL   #1                                                           ; 0946
                    00000000'       EF  9F 001D1          PUSHAB  DBG$CS_MODIFY                                                ;
                                    53  DD 001D7          PUSHL   R3                                                           ;
                00000000G   00      03  FB 001D9          CALLS   #3, DBG$NMATCH                                               ;
                            01      50  D1 001E0          CMPL    R0, #1                                                       ;
```

```
                               10  12 001E3           BNEQ    30$                          ;
                               56  D5 001E5           TSTL    CMD_TYPE                     ; 0953
                               05  13 001E7           BEQL    28$                          ;
               01              56  D1 001E9           CMPL    CMD_TYPE, #1                 ; 0954
                               48  12 001EC           BNEQ    35$                          ;
                               57  D4 001EE 28$:      CLRL    CMD_KIND                     ; 0957
               55              02  D0 001F0           MOVL    #2, SUB_KIND                 ; 0958
                               75  11 001F3 29$:      BRB     40$                          ; 0953
                               01  DD 001F5 30$:      PUSHL   #1                           ; 0969
                   00000000'   EF  9F 001F7           PUSHAB  DBG$CS_EXECUTE               ;
                               53  DD 001FD           PUSHL   R3                           ;
    00000000G  00              03  FB 001FF           CALLS   #3, DBG$NMATCH               ;
               01              50  D1 00206           CMPL    R0, #1                       ;
                               10  12 00209           BNEQ    34$                          ;
                               56  D5 0020B           TSTL    CMD_TYPE                     ; 0976
                               05  13 0020D           BEQL    32$                          ;
               01              56  D1 0020F           CMPL    CMD_TYPE, #1                 ; 0977
                               22  12 00212 31$:      BNEQ    35$                          ;
                               57  D4 00214 32$:      CLRL    CMD_KIND                     ; 0980
               55              03  D0 00216           MOVL    #3, SUB_KIND                 ; 0981
                               6E  11 00219 33$:      BRB     42$                          ; 0976
                               01  DD 0021B 34$:      PUSHL   #1                           ; 0992
                   00000000'   EF  9F 0021D           PUSHAB  DBG$CS_RETURN                ;
                               53  DD 00223           PUSHL   R3                           ;
    00000000G  00              03  FB 00225           CALLS   #3, DBG$NMATCH               ;
               01              50  D1 0022C           CMPL    R0, #1                       ;
                               0F  12 0022F           BNEQ    38$                          ;
               02              56  D1 00231           CMPL    CMD_TYPE, #2                 ; 0999
                               03  12 00234           BNEQ    36$                          ;
                           07FF    31 00236 35$:      BRW     138$                         ;
                               57  D4 00239 36$:      CLRL    CMD_KIND                     ; 1002
               55              04  D0 0023B           MOVL    #4, SUB_KIND                 ; 1003
                               65  11 0023E 37$:      BRB     44$                          ; 0999
                               01  DD 00240 38$:      PUSHL   #1                           ; 1014
                   00000000'   EF  9F 00242           PUSHAB  DBG$CS_EXCEPTION             ;
                               53  DD 00248           PUSHL   R3                           ;
    00000000G  00              03  FB 0024A           CALLS   #3, DBG$NMATCH               ;
               01              50  D1 00251           CMPL    R0, #1                       ;
                               16  12 00254           BNEQ    41$                          ;
                               56  D5 00256           TSTL    CMD_TYPE                     ; 1021
                               0A  13 00258           BEQL    39$                          ;
               01              56  D1 0025A           CMPL    CMD_TYPE, #1                 ; 1022
                               05  13 0025D           BEQL    39$                          ;
               03              56  D1 0025F           CMPL    CMD_TYPE, #3                 ; 1023
                               D2  12 00262           BNEQ    35$                          ;
               57              02  D0 00264 39$:      MOVL    #2, CMD_KIND                 ; 1026
               55              0F  D0 00267           MOVL    #15, SUB_KIND                ; 1027
                               7E  11 0026A 40$:      BRB     47$                          ; 1021
                               02  DD 0026C 41$:      PUSHL   #2                           ; 1038
                   00000000'   EF  9F 0026E           PUSHAB  DBG$CS_SOURCE                ;
                               53  DD 00274           PUSHL   R3                           ;
    00000000G  00              03  FB 00276           CALLS   #3, DBG$NMATCH               ;
               01              50  D1 0027D           CMPL    R0, #1                       ;
                               09  12 00280           BNEQ    43$                          ;
           14  AE              01  D0 00282           MOVL    #1, SOURCE                   ; 1040
                           10  AE  D4 00286           CLRL    NOSOURCE                     ; 1041
                               5F  11 00289 42$:      BRB     47$                          ; 0820
```

```
                                04  DD 0028B 43$:    PUSHL   #4                                       ;1047
                    00000000'   EF  9F 0028D          PUSHAB  DBG$CS_NOSOURCE                         ;
                                53  DD 00293          PUSHL   R3                                      ;
00000000G   00                  03  FB 00295          CALLS   #3, DBG$NMATCH                          ;
            01                  50  D1 0029C          CMPL    R0, #1                                  ;
                                06  12 0029F          BNEQ    45$                                     ;
         10 AE                  01  7D 002A1          MOVQ    #1, NOSOURCE                            ;1049
                                66  11 002A5 44$:      BRB     49$                                     ;0820
                                02  DD 002A7 45$:      PUSHL   #2                                      ;1056
                    00000000'   EF  9F 002A9          PUSHAB  DBG$CS_SYSTEM                           ;
                                53  DD 002AF          PUSHL   R3                                      ;
00000000G   00                  03  FB 002B1          CALLS   #3, DBG$NMATCH                          ;
            01                  50  D1 002B8          CMPL    R0, #1                                  ;
                                0E  12 002BB          BNEQ    46$                                     ;
            02                  56  D1 002BD          CMPL    CMD_TYPE, #2                            ;1063
                                66  13 002C0          BEQL    51$                                     ;
         0C AE                  01  D0 002C2          MOVL    #1, SYSTEM                              ;1066
                 08 AE          D4  002C6          CLRL    NOSYSTEM                                ;1067
                                62  11 002C9          BRB     52$                                     ;1063
                                04  DD 002CB 46$:      PUSHL   #4                                      ;1078
                    00000000'   EF  9F 002CD          PUSHAB  DBG$CS_NOSYSTEM                         ;
                                53  DD 002D3          PUSHL   R3                                      ;
00000000G   00                  03  FB 002D5          CALLS   #3, DBG$NMATCH                          ;
            01                  50  D1 002DC          CMPL    R0, #1                                  ;
                                0B  12 002DF          BNEQ    48$                                     ;
            02                  56  D1 002E1          CMPL    CMD_TYPE, #2                            ;1085
                                62  13 002E4          BEQL    54$                                     ;
         08 AE                  01  7D 002E6          MOVQ    #1, NOSYSTEM                            ;1088
                                64  11 002EA 47$:      BRB     55$                                     ;1085
                                01  DD 002EC 48$:      PUSHL   #1                                      ;1100
                    00000000'   EF  9F 002EE          PUSHAB  DBG$CS_INTO                             ;
                                53  DD 002F4          PUSHL   R3                                      ;
00000000G   00                  03  FB 002F6          CALLS   #3, DBG$NMATCH                          ;
            01                  50  D1 002FD          CMPL    R0, #1                                  ;
                                0D  12 00300          BNEQ    50$                                     ;
            02                  56  D1 00302          CMPL    CMD_TYPE, #2                            ;1107
                                64  13 00305          BEQL    57$                                     ;
         04 AE                  01  D0 00307          MOVL    #1, INTO                                ;1110
                                6E  D4 0030B          CLRL    OVER                                    ;1111
                                64  11 0030D 49$:      BRB     58$                                     ;1107
                                01  DD 0030F 50$:      PUSHL   #1                                      ;1121
                    00000000'   EF  9F 00311          PUSHAB  DBG$CS_OVER                             ;
                                53  DD 00317          PUSHL   R3                                      ;
00000000G   00                  03  FB 00319          CALLS   #3, DBG$NMATCH                          ;
            01                  50  D1 00320          CMPL    R0, #1                                  ;
                                0A  12 00323          BNEQ    53$                                     ;
            02                  56  D1 00325          CMPL    CMD_TYPE, #2                            ;1128
                                64  13 00328 51$:      BEQL    60$                                     ;
            6E                  01  7D 0032A          MOVQ    #1, OVER                                ;1131
                                67  11 0032D 52$:      BRB     61$                                     ;1128
                                01  DD 0032F 53$:      PUSHL   #1                                      ;1143
                    00000000'   EF  9F 00331          PUSHAB  DBG$CS_CALL                             ;
                                53  DD 00337          PUSHL   R3                                      ;
00000000G   00                  03  FB 00339          CALLS   #3, DBG$NMATCH                          ;
            01                  50  D1 00340          CMPL    R0, #1                                  ;
                                0D  12 00343          BNEQ    56$                                     ;
            02                  56  D1 00345          CMPL    CMD_TYPE, #2                            ;1150
```

```
                                 6B  13 00348 54$:    BEQL    64$                                  ;
              57                 01  D0 0034A         MOVL    #1, CMD_KIND                         ; 1153
              55                 05  D0 0034D         MOVL    #5, SUB_KIND                         ; 1154
                                 44  11 00350 55$:    BRB     61$                                  ; 1150
                                 01  DD 00352 56$:    PUSHL   #1                                   ; 1165
                    00000000'    EF  9F 00354         PUSHAB  DBG$CS_BRANCH                        ;
                                 53  DD 0035A         PUSHL   R3                                   ;
    00000000G     00             03  FB 0035C         CALLS   #3, DBG$NMATCH                       ;
                  01             50  D1 00363         CMPL    R0, #1                               ;
                                 0D  12 00366         BNEQ    59$                                  ;
                  02             56  D1 00368         CMPL    CMD_TYPE, #2                         ; 1172
                                 48  13 0036B 57$:    BEQL    64$                                  ;
              57                 01  D0 0036D         MOVL    #1, CMD_KIND                         ; 1175
              55                 06  D0 00370         MOVL    #6, SUB_KIND                         ; 1176
                                 21  11 00373 58$:    BRB     61$                                  ; 1172
                                 01  DD 00375 59$:    PUSHL   #1                                   ; 1187
                    00000000'    EF  9F 00377         PUSHAB  DBG$CS_LINE                          ;
                                 53  DD 0037D         PUSHL   R3                                   ;
    00000000G     00             03  FB 0037F         CALLS   #3, DBG$NMATCH                       ;
                  01             50  D1 00386         CMPL    R0, #1                               ;
                                 0E  12 00389         BNEQ    62$                                  ;
                  02             56  D1 0038B         CMPL    CMD_TYPE, #2                         ; 1194
                                 25  13 0038E 60$:    BEQL    64$                                  ;
              57                 01  D0 00390         MOVL    #1, CMD_KIND                         ; 1197
              55                 07  D0 00393         MOVL    #7, SUB_KIND                         ; 1198
                               FD0B  31 00396 61$:    BRW     10$                                  ; 1194
                                 01  DD 00399 62$:    PUSHL   #1                                   ; 1210
                    00000000'    EF  9F 0039B         PUSHAB  DBG$CS_INSTRUCTION                   ;
                                 53  DD 003A1         PUSHL   R3                                   ;
    00000000G     00             03  FB 003A3         CALLS   #3, DBG$NMATCH                       ;
                  01             50  D1 003AA         CMPL    R0, #1                               ;
                                 03  13 003AD         BEQL    63$                                  ;
                               00A7  31 003AF         BRW     68$                                  ;
                  02             56  D1 003B2 63$:    CMPL    CMD_TYPE, #2                         ; 1217
                                 7B  13 003B5 64$:    BEQL    66$                                  ;
              57                 01  D0 003B7         MOVL    #1, CMD_KIND                         ; 1220
              55                 08  D0 003BA         MOVL    #8, SUB_KIND                         ; 1221
                                 01  DD 003BD         PUSHL   #1                                   ; 1226
                    00000000'    EF  9F 003BF         PUSHAB  DBG$CS_EQUALS                        ;
                                 53  DD 003C5         PUSHL   R3                                   ;
    00000000G     00             03  FB 003C7         CALLS   #3, DBG$NMATCH                       ;
                  C5             50  E9 003CE         BLBC    R0, 61$                              ;
                  55             09  D0 003D1         MOVL    #9, SUB_KIND                         ; 1232
                                 01  DD 003D4         PUSHL   #1                                   ; 1237
                    00000000'    EF  9F 003D6         PUSHAB  DBG$CS_L_PAREN                       ;
                                 53  DD 003DC         PUSHL   R3                                   ;
    00000000G     00             03  FB 003DE         CALLS   #3, DBG$NMATCH                       ;
                  4D             50  E9 003E5         BLBC    R0, 67$                              ;
                                 04  DD 003E8 65$:    PUSHL   #4                                   ; 1245
    00000000G     00             01  FB 003EA         CALLS   #1, DBG$GET_TEMPMEM                  ;
                  54             50  D0 003F1         MOVL    R0, NOUN_NODE                        ;
                         08      A4  D4 003F4         CLRL    8(NOUN_NODE)                         ;
                  69             54  D0 003F7         MOVL    NOUN_NODE, (NOUN_LINK)               ;
                  59     08      A4  9E 003FA         MOVAB   8(NOUN_NODE), NOUN_LINK              ;
                                 53  DD 003FE         PUSHL   R3                                   ; 1253
    00000000G     00             01  FB 00400         CALLS   #1, DBG$OPCODE_INDEX                 ;
                  64             50  D0 00407         MOVL    R0, (NOUN_NODE)                      ;
```

```
                                01   DD 0040A          PUSHL   #1                                      ; 1256
                    00000000'   EF   9F 0040C          PUSHAB  DBG$CS_COMMA
                                53   DD 00412          PUSHL   R3
00000000G     00                03   FB 00414          CALLS   #3, DBG$NMATCH
              CA                50   E8 0041B          BLBS    R0, 65$
                                01   DD 0041E          PUSHL   #1                                      ; 1261
                    00000000'   EF   9F 00420          PUSHAB  DBG$CS_R_PAREN
                                53   DD 00426          PUSHL   R3
00000000G     00                03   FB 00428          CALLS   #3, DBG$NMATCH
              70                50   E8 0042F          BLBS    R0, 71$
                          0603  31   00432 66$:        BRW     138$
                                04   DD 00435 67$:     PUSHL   #4                                      ; 1265
00000000G     00                01   FB 00437          CALLS   #1, DBG$GET_TEMPMEM
              54                50   D0 0043E          MOVL    R0, NOUN_NODE
                          08    A4   D4 00441          CLRL    8(NOUN_NODE)
              69                54   D0 00444          MOVL    NOUN_NODE, (NOUN_LINK)
              59          08    A4   9E 00447          MOVAB   8(NOUN_NODE), NOUN_LINK
                                53   DD 0044B          PUSHL   R3                                      ; 1272
00000000G     00                01   FB 0044D          CALLS   #1, DBG$OPCODE_INDEX
              64                50   D0 00454          MOVL    R0, (NOUN_NODE)
                                49   11 00457          BRB     71$                                     ; 1226
                                01   DD 00459 68$:     PUSHL   #1                                      ; 1287
                    00000000'   EF   9F 0045B          PUSHAB  P.ABD
                                53   DD 00461          PUSHL   R3
00000000G     00                03   FB 00463          CALLS   #3, DBG$NMATCH
              01                50   D1 0046A          CMPL    R0, #1
                                08   12 0046D          BNEQ    69$
                    00000000'   EF   9F 0046F          PUSHAB  P.ABE                                   ; 1288
                                1C   11 00475          BRB     70$
                                03   DD 00477 69$:     PUSHL   #3                                      ; 1290
                    00000000'   EF   9F 00479          PUSHAB  P.ABF
                                53   DD 0047F          PUSHL   R3
00000000G     00                03   FB 00481          CALLS   #3, DBG$NMATCH
              01                50   D1 00488          CMPL    R0, #1
                                A5   12 0048B          BNEQ    66$
                    00000000'   EF   9F 0048D          PUSHAB  P.ABG                                   ; 1291
                                01   DD 00493 70$:     PUSHL   #1
                    00028148    8F   DD 00495          PUSHL   #164168
00000000G     00                03   FB 00498          CALLS   #3, LIB$SIGNAL
                          FBFF  31   004A2 71$:        BRW     10$
                                01   DD 004A5 72$:     PUSHL   #1                                      ; 1329
                    00000000'   EF   9F 004A7          PUSHAB  DBG$CS_RETURN
                                53   DD 004AD          PUSHL   R3
00000000G     00                03   FB 004AF          CALLS   #3, DBG$NMATCH
              01                50   D1 004B6          CMPL    R0, #1
                                07   12 004B9          BNEQ    73$
                                57   D4 004BB          CLRL    CMD_KIND                                ; 1331
              55                04   D0 004BD          MOVL    #4, SUB_KIND                            ; 1332
                                74   11 004C0          BRB     77$                                     ; 1323
                                02   DD 004C2 73$:     PUSHL   #2                                      ; 1338
                    00000000'   EF   9F 004C4          PUSHAB  DBG$CS_SOURCE
                                53   DD 004CA          PUSHL   R3
00000000G     00                03   FB 004CC          CALLS   #3, DBG$NMATCH
              01                50   D1 004D3          CMPL    R0, #1
                                09   12 004D6          BNEQ    74$
              14          AE    01   D0 004D8          MOVL    #1, SOURCE                              ; 1340
                    10    AE    D4   D4 004DC          CLRL    NOSOURCE                                ; 1341
```

```
                                72  11  004DF         BRB     79$                             ;  1323
                                04  DD  004E1 74$:    PUSHL   #4                              ;  1347
                    00000000'   EF  9F  004E3         PUSHAB  DBG$CS_NOSOURCE                 ;
                                53  DD  004E9         PUSHL   R3                              ;
00000000G   00                  03  FB  004EB         CALLS   #3, DBG$NMATCH                  ;
            01                  50  D1  004F2         CMPL    R0, #1                          ;
                                06  12  004F5         BNEQ    75$                             ;
      10    AE                  01  7D  004F7         MOVQ    #1, NOSOURCE                    ;  1349
                                71  11  004FB         BRB     81$                             ;  1323
                                02  DD  004FD 75$:    PUSHL   #2                              ;  1356
                    00000000'   EF  9F  004FF         PUSHAB  DBG$CS_SYSTEM                   ;
                                53  DD  00505         PUSHL   R3                              ;
00000000G   00                  03  FB  00507         CALLS   #3, DBG$NMATCH                  ;
            01                  50  D1  0050E         CMPL    R0, #1                          ;
                                09  12  00511         BNEQ    76$                             ;
      0C    AE                  01  D0  00513         MOVL    #1, SYSTEM                      ;  1358
                        08      AE  D4  00517         CLRL    NOSYSTEM                        ;  1359
                                70  11  0051A         BRB     83$                             ;  1323
                                04  DD  0051C 76$:    PUSHL   #4                              ;  1365
                    00000000'   EF  9F  0051E         PUSHAB  DBG$CS_NOSYSTEM                 ;
                                53  DD  00524         PUSHL   R3                              ;
00000000G   00                  03  FB  00526         CALLS   #3, DBG$NMATCH                  ;
            01                  50  D1  0052D         CMPL    R0, #1                          ;
                                06  12  00530         BNEQ    78$                             ;
      08    AE                  01  7D  00532         MOVQ    #1, NOSYSTEM                    ;  1367
                                6F  11  00536 77$:    BRB     85$                             ;  1323
                                02  DD  00538 78$:    PUSHL   #2                              ;  1374
                    00000000'   EF  9F  0053A         PUSHAB  DBG$CS_SILENT                   ;
                                53  DD  00540         PUSHL   R3                              ;
00000000G   00                  03  FB  00542         CALLS   #3, DBG$NMATCH                  ;
            01                  50  D1  00549         CMPL    R0, #1                          ;
                                07  12  0054C         BNEQ    80$                             ;
            5B                  01  D0  0054E         MOVL    #1, SILENT                      ;  1376
                                5A  D4  00551         CLRL    NOSILENT                        ;  1377
                                70  11  00553 79$:    BRB     87$                             ;  1323
                                04  DD  00555 80$:    PUSHL   #4                              ;  1383
                    00000000'   EF  9F  00557         PUSHAB  DBG$CS_NOSILENT                 ;
                                53  DD  0055D         PUSHL   R3                              ;
00000000G   00                  03  FB  0055F         CALLS   #3, DBG$NMATCH                  ;
            01                  50  D1  00566         CMPL    R0, #1                          ;
                                05  12  00569         BNEQ    82$                             ;
            5A                  01  7D  0056B         MOVQ    #1, NOSILENT                    ;  1386
                                73  11  0056E 81$:    BRB     89$                             ;  1323
                                01  DD  00570 82$:    PUSHL   #1                              ;  1392
                    00000000'   EF  9F  00572         PUSHAB  DBG$CS_INTO                     ;
                                53  DD  00578         PUSHL   R3                              ;
00000000G   00                  03  FB  0057A         CALLS   #3, DBG$NMATCH                  ;
            01                  50  D1  00581         CMPL    R0, #1                          ;
                                08  12  00584         BNEQ    84$                             ;
      04    AE                  01  D0  00586         MOVL    #1, INTO                        ;  1394
                                6E  D4  0058A         CLRL    OVER                            ;  1395
                                73  11  0058C 83$:    BRB     91$                             ;  1323
                                01  DD  0058E 84$:    PUSHL   #1                              ;  1401
                    00000000'   EF  9F  00590         PUSHAB  DBG$CS_OVER                     ;
                                53  DD  00596         PUSHL   R3                              ;
00000000G   00                  03  FB  00598         CALLS   #3, DBG$NMATCH                  ;
            01                  50  D1  0059F         CMPL    R0, #1                          ;
```

```
                                      05  12 005A2        BNEQ    86$                         ;
                  6E                  01  7D 005A4        MOVQ    #1, OVER                    ; 1403
                                      76  11 005A7 85$:   BRB     93$                         ; 1323
                                      01  DD 005A9 86$:   PUSHL   #1                          ; 1410
                          00000000'   EF  9F 005AB        PUSHAB  DBG$CS_CALL                 ;
                                      53  DD 005B1        PUSHL   R3                          ;
          00000000G  00               03  FB 005B3        CALLS   #3, DBG$NMATCH              ;
                     01               50  D1 005BA        CMPL    R0, #1                      ;
                                      08  12 005BD        BNEQ    88$                         ;
                     57               01  D0 005BF        MOVL    #1, CMD_KIND                ; 1412
                     55               05  D0 005C2        MOVL    #5, SUB_KIND                ; 1413
                                      58  11 005C5 87$:   BRB     93$                         ; 1323
                                      01  DD 005C7 88$:   PUSHL   #1                          ; 1419
                          00000000'   EF  9F 005C9        PUSHAB  DBG$CS_BRANCH               ;
                                      53  DD 005CF        PUSHL   R3                          ;
          00000000G  00               03  FB 005D1        CALLS   #3, DBG$NMATCH              ;
                     01               50  D1 005D8        CMPL    R0, #1                      ;
                                      08  12 005DB        BNEQ    90$                         ;
                     57               01  D0 005DD        MOVL    #1, CMD_KIND                ; 1421
                     55               06  D0 005E0        MOVL    #6, SUB_KIND                ; 1422
                                      3A  11 005E3 89$:   BRB     93$                         ; 1323
                                      01  DD 005E5 90$:   PUSHL   #1                          ; 1428
                          00000000'   EF  9F 005E7        PUSHAB  DBG$CS_EXCEPTION            ;
                                      53  DD 005ED        PUSHL   R3                          ;
          00000000G  00               03  FB 005EF        CALLS   #3, DBG$NMATCH              ;
                     01               50  D1 005F6        CMPL    R0, #1                      ;
                                      08  12 005F9        BNEQ    92$                         ;
                     57               02  D0 005FB        MOVL    #2, CMD_KIND                ; 1430
                     55               0F  D0 005FE        MOVL    #15, SUB_KIND               ; 1431
                                      1C  11 00601 91$:   BRB     93$                         ; 1323
                                      01  DD 00603 92$:   PUSHL   #1                          ; 1437
                          00000000'   EF  9F 00605        PUSHAB  DBG$CS_LINE                 ;
                                      53  DD 0060B        PUSHL   R3                          ;
          00000000G  00               03  FB 0060D        CALLS   #3, DBG$NMATCH              ;
                     01               50  D1 00614        CMPL    R0, #1                      ;
                                      09  12 00617        BNEQ    94$                         ;
                     57               01  D0 00619        MOVL    #1, CMD_KIND                ; 1439
                     55               07  D0 0061C        MOVL    #7, SUB_KIND                ; 1440
                                    0104  31 0061F 93$:   BRW     102$                        ; 1323
                                      01  DD 00622 94$:   PUSHL   #1                          ; 1447
                          00000000'   EF  9F 00624        PUSHAB  DBG$CS_INSTRUCTION          ;
                                      53  DD 0062A        PUSHL   R3                          ;
          00000000G  00               03  FB 0062C        CALLS   #3, DBG$NMATCH              ;
                     01               50  D1 00633        CMPL    R0, #1                      ;
                                      03  13 00636        BEQL    95$                         ;
                                    00A2  31 00638        BRW     99$                         ;
                     57               01  D0 0063B 95$:   MOVL    #1, CMD_KIND                ; 1449
                     55               08  D0 0063E        MOVL    #8, SUB_KIND                ; 1450
                                      01  DD 00641        PUSHL   #1                          ; 1455
                          00000000'   EF  9F 00643        PUSHAB  DBG$CS_EQUALS               ;
                                      53  DD 00649        PUSHL   R3                          ;
          00000000G  00               03  FB 0064B        CALLS   #3, DBG$NMATCH              ;
                     CA               50  E9 00652        BLBC    R0, 93$                     ;
                     55               09  D0 00655        MOVL    #9, SUB_KIND                ; 1461
                                      01  DD 00658        PUSHL   #1                          ; 1465
                          00000000'   EF  9F 0065A        PUSHAB  DBG$CS_L_PAREN              ;
                                      53  DD 00660        PUSHL   R3                          ;
```

```
00000000G  00             03  FB 00662         CALLS   #3, DBG$NMATCH                 ;
           4D             50  E9 00669         BLBC    R0, 98$                        ;
                          04  DD 0066C  96$:   PUSHL   #4                             ; 1472
00000000G  00             01  FB 0066E         CALLS   #1, DBG$GET_TEMPMEM            ;
           54             50  D0 00675         MOVL    R0, NOUN_NODE                  ;
                     08   A4  D4 00678         CLRL    8(NOUN_NODE)                   ;
           69             54  D0 0067B         MOVL    NOUN_NODE, (NOUN_LINK)         ;
           59        08   A4  9E 0067E         MOVAB   8(NOUN_NODE), NOUN_LINK        ;
                          53  DD 00682         PUSHL   R3                             ; 1479
00000000G  00             01  FB 00684         CALLS   #1, DBG$OPCODE_INDEX           ;
           64             50  D0 0068B         MOVL    R0, (NOUN_NODE)                ;
                          01  DD 0068E         PUSHL   #1                             ; 1482
              00000000'   EF  9F 00690         PUSHAB  DBG$CS_COMMA                   ;
                          53  DD 00696         PUSHL   R3                             ;
00000000G  00             03  FB 00698         CALLS   #3, DBG$NMATCH                 ;
           CA             50  E8 0069F         BLBS    R0, 96$                        ;
                          01  DD 006A2         PUSHL   #1                             ; 1487
              00000000'   EF  9F 006A4         PUSHAB  DBG$CS_R_PAREN                 ;
                          53  DD 006AA         PUSHL   R3                             ;
00000000G  00             03  FB 006AC         CALLS   #3, DBG$NMATCH                 ;
           70             50  E8 006B3         BLBS    R0, 102$                       ;
                        037F  31 006B6  97$:   BRW     138$                           ;
                          04  DD 006B9  98$:   PUSHL   #4                             ; 1491
00000000G  00             01  FB 006BB         CALLS   #1, DBG$GET_TEMPMEM            ;
           54             50  D0 006C2         MOVL    R0, NOUN_NODE                  ;
                     08   A4  D4 006C5         CLRL    8(NOUN_NODE)                   ;
           69             54  D0 006C8         MOVL    NOUN_NODE, (NOUN_LINK)         ;
           59        08   A4  9E 006CB         MOVAB   8(NOUN_NODE), NOUN_LINK        ;
                          53  DD 006CF         PUSHL   R3                             ; 1497
00000000G  00             01  FB 006D1         CALLS   #1, DBG$OPCODE_INDEX           ;
           64             50  D0 006D8         MOVL    R0, (NOUN_NODE)                ;
                          49  11 006DB         BRB     102$                           ; 1455
                          01  DD 006DD  99$:   PUSHL   #1                             ; 1507
              00000000'   EF  9F 006DF         PUSHAB  P.ABH                          ;
                          53  DD 006E5         PUSHL   R3                             ;
00000000G  00             03  FB 006E7         CALLS   #3, DBG$NMATCH                 ;
           01             50  D1 006EE         CMPL    R0, #1                         ;
                          08  12 006F1         BNEQ    100$                           ;
              00000000'   EF  9F 006F3         PUSHAB  P.ABI                          ; 1508
                          1C  11 006F9         BRB     101$                           ;
                          03  DD 006FB  100$:  PUSHL   #3                             ; 1510
              00000000'   EF  9F 006FD         PUSHAB  P.ABJ                          ;
                          53  DD 00703         PUSHL   R3                             ;
00000000G  00             03  FB 00705         CALLS   #3, DBG$NMATCH                 ;
           01             50  D1 0070C         CMPL    R0, #1                         ;
                          A5  12 0070F         BNEQ    97$                            ;
              00000000'   EF  9F 00711         PUSHAB  P.ABK                          ; 1511
                          01  DD 00717  101$:  PUSHL   #1                             ;
              00028148    8F  DD 00719         PUSHL   #164168                        ;
00000000G  00             03  FB 0071F         CALLS   #3, LIB$SIGNAL                 ;
                          01  DD 00726  102$:  PUSHL   #1                             ; 1523
              00000000'   EF  9F 00728         PUSHAB  DBG$CS_COMMA                   ;
                          53  DD 0072E         PUSHL   R3                             ;
00000000G  00             03  FB 00730         CALLS   #3, DBG$NMATCH                 ;
           03             50  E9 00737         BLBC    R0, 103$                       ;
                        FD68  31 0073A         BRW     72$                            ;
                          03  DD 0073D  103$:  PUSHL   #3                             ; 1525
```

```
                00000000G  00              01  FB  0073F          CALLS   #1, DBG$GET_TEMPMEM                      ;
                           52              50  D0  00746          MOVL    R0, ADVERB_NODE                         ;
                                     08    A2  D4  00749          CLRL    8(ADVERB_NODE)                          ;
                           68              52  D0  0074C          MOVL    ADVERB_NODE, (ADVERB_LINK)              ;
                           58        08    A2  9E  0074F          MOVAB   8(ADVERB_NODE), ADVERB_LINK             ;
                                           57  D5  00753          TSTL    CMD_KIND                                ; 1538
                                           31  12  00755          BNEQ    108$                                    ;
                                           55  D5  00757          TSTL    SUB_KIND                                ; 1542
                                           05  12  00759          BNEQ    104$                                    ;
                                50         06  D0  0075B          MOVL    #6, R0                                  ;
                                           6C  11  0075E          BRB     116$                                    ;
                                01         55  D1  00760  104$:   CMPL    SUB_KIND, #1                            ; 1544
                                           05  12  00763          BNEQ    105$                                    ;
                                50         07  D0  00765          MOVL    #7, R0                                  ;
                                           62  11  00768          BRB     116$                                    ;
                                02         55  D1  0076A  105$:   CMPL    SUB_KIND, #2                            ; 1546
                                           05  12  0076D          BNEQ    106$                                    ;
                                50         08  D0  0076F          MOVL    #8, R0                                  ;
                                           58  11  00772          BRB     116$                                    ;
                                03         55  D1  00774  106$:   CMPL    SUB_KIND, #3                            ; 1548
                                           05  12  00777          BNEQ    107$                                    ;
                                50         09  D0  00779          MOVL    #9, R0                                  ;
                                           4E  11  0077C          BRB     116$                                    ;
                                04         55  D1  0077E  107$:   CMPL    SUB_KIND, #4                            ; 1550
                                           41  12  00781          BNEQ    114$                                    ;
                                50         0A  D0  00783          MOVL    #10, R0                                 ;
                                           44  11  00786          BRB     116$                                    ; 1540
                                02         57  D1  00788  108$:   CMPL    CMD_KIND, #2                            ; 1556
                                           05  12  0078B          BNEQ    109$                                    ;
                                62         0B  90  0078D          MOVB    #11, (ADVERB_NODE)                      ; 1557
                                           3D  11  00790          BRB     117$                                    ;
                                01         57  D1  00792  109$:   CMPL    CMD_KIND, #1                            ; 1560
                                           38  12  00795          BNEQ    117$                                    ;
                                05         55  D1  00797          CMPL    SUB_KIND, #5                            ; 1564
                                           05  12  0079A          BNEQ    110$                                    ;
                                50         12  D0  0079C          MOVL    #18, R0                                 ;
                                           2B  11  0079F          BRB     116$                                    ;
                                06         55  D1  007A1  110$:   CMPL    SUB_KIND, #6                            ; 1566
                                           05  12  007A4          BNEQ    111$                                    ;
                                50         13  D0  007A6          MOVL    #19, R0                                 ;
                                           21  11  007A9          BRB     116$                                    ;
                                08         55  D1  007AB  111$:   CMPL    SUB_KIND, #8                            ; 1568
                                           05  12  007AE          BNEQ    112$                                    ;
                                50         15  D0  007B0          MOVL    #21, R0                                 ;
                                           17  11  007B3          BRB     116$                                    ;
                                09         55  D1  007B5  112$:   CMPL    SUB_KIND, #9                            ; 1570
                                           05  12  007B8          BNEQ    113$                                    ;
                                50         16  D0  007BA          MOVL    #22, R0                                 ;
                                           0D  11  007BD          BRB     116$                                    ;
                                07         55  D1  007BF  113$:   CMPL    SUB_KIND, #7                            ; 1572
                                           05  13  007C2          BEQL    115$                                    ;
                                50         01  CE  007C4  114$:   MNEGL   #1, R0                                  ;
                                           03  11  007C7          BRB     116$                                    ;
                                50         14  D0  007C9  115$:   MOVL    #20, R0                                 ;
                           62              50  90  007CC  116$:   MOVB    R0, (ADVERB_NODE)                       ; 1562
                           01        20    AE  D1  007CF  117$:   CMPL    AFTER_COUNT, #1                         ; 1583
                                           27  13  007D3          BEQL    119$                                    ;
```

```
                         03  DD 007D5           PUSHL   #3                                      ; 1585
00000000G   00           01  FB 007D7           CALLS   #1, DBG$GET_TEMPMEM                     ;
            52           50  D0 007DE           MOVL    R0, ADVERB_NODE                         ;
                    08   A2  D4 007E1           CLRL    8(ADVERB_NODE)                          ;
            68           52  D0 007E4           MOVL    ADVERB_NODE, (ADVERB_LINK)              ;
            58      08   A2  9E 007E7           MOVAB   8(ADVERB_NODE), ADVERB_LINK             ;
                    20   AE  D5 007EB           TSTL    AFTER_COUNT                             ; 1592
                         05  12 007EE           BNEQ    118$                                    ;
            62           01  90 007F0           MOVB    #1, (ADVERB_NODE)                       ; 1594
                         07  11 007F3           BRB     119$                                    ;
                         62  94 007F5 118$:     CLRB    (ADVERB_NODE)                           ; 1597
               04   A2       20  AE D0 007F7    MOVL    AFTER_COUNT, 4(ADVERB_NODE)             ; 1598
            1B           5B  E9 007FC 119$:     BLBC    SILENT, 120$                            ; 1606
                         03  DD 007FF           PUSHL   #3                                      ; 1608
00000000G   00           01  FB 00801           CALLS   #1, DBG$GET_TEMPMEM                     ;
            52           50  D0 00808           MOVL    R0, ADVERB_NODE                         ;
                    08   A2  D4 0080B           CLRL    8(ADVERB_NODE)                          ;
            68           52  D0 0080E           MOVL    ADVERB_NODE, (ADVERB_LINK)              ;
            58      08   A2  9E 00811           MOVAB   8(ADVERB_NODE), ADVERB_LINK             ;
            62           02  90 00815           MOVB    #2, (ADVERB_NODE)                       ; 1610
                         1C  11 00818           BRB     121$                                    ; 1606
            19           5A  E9 0081A 120$:     BLBC    NOSILENT, 121$                          ; 1613
                         03  DD 0081D           PUSHL   #3                                      ; 1615
00000000G   00           01  FB 0081F           CALLS   #1, DBG$GET_TEMPMEM                     ;
            52           50  D0 00826           MOVL    R0, ADVERB_NODE                         ;
                    08   A2  D4 00829           CLRL    8(ADVERB_NODE)                          ;
            68           52  D0 0082C           MOVL    ADVERB_NODE, (ADVERB_LINK)              ;
            58      08   A2  9E 0082F           MOVAB   8(ADVERB_NODE), ADVERB_LINK             ;
            62           03  90 00833           MOVB    #3, (ADVERB_NODE)                       ; 1617
            19      18   AE  E9 00836 121$:     BLBC    TEMPORARY, 122$                         ; 1623
                         03  DD 0083A           PUSHL   #3                                      ; 1625
00000000G   00           01  FB 0083C           CALLS   #1, DBG$GET_TEMPMEM                     ;
            52           50  D0 00843           MOVL    R0, ADVERB_NODE                         ;
                    08   A2  D4 00846           CLRL    8(ADVERB_NODE)                          ;
            68           52  D0 00849           MOVL    ADVERB_NODE, (ADVERB_LINK)              ;
            58      08   A2  9E 0084C           MOVAB   8(ADVERB_NODE), ADVERB_LINK             ;
            62           01  90 00850           MOVB    #1, (ADVERB_NODE)                       ; 1627
            19      14   AE  E9 00853 122$:     BLBC    SOURCE, 123$                            ; 1634
                         03  DD 00857           PUSHL   #3                                      ; 1636
00000000G   00           01  FB 00859           CALLS   #1, DBG$GET_TEMPMEM                     ;
            52           50  D0 00860           MOVL    R0, ADVERB_NODE                         ;
                    08   A2  D4 00863           CLRL    8(ADVERB_NODE)                          ;
            68           52  D0 00866           MOVL    ADVERB_NODE, (ADVERB_LINK)              ;
            58      08   A2  9E 00869           MOVAB   8(ADVERB_NODE), ADVERB_LINK             ;
            62           0C  90 0086D           MOVB    #12, (ADVERB_NODE)                      ; 1638
            19      10   AE  E9 00870 123$:     BLBC    NOSOURCE, 124$                          ; 1641
                         03  DD 00874           PUSHL   #3                                      ; 1643
00000000G   00           01  FB 00876           CALLS   #1, DBG$GET_TEMPMEM                     ;
            52           50  D0 0087D           MOVL    R0, ADVERB_NODE                         ;
                    08   A2  D4 00880           CLRL    8(ADVERB_NODE)                          ;
            68           52  D0 00883           MOVL    ADVERB_NODE, (ADVERB_LINK)              ;
            58      08   A2  9E 00886           MOVAB   8(ADVERB_NODE), ADVERB_LINK             ;
            62           0D  90 0088A           MOVB    #13, (ADVERB_NODE)                      ; 1645
            01           57  D1 0088D 124$:     CMPL    CMD_KIND, #1                            ; 1651
                         09  13 00890           BEQL    125$                                    ;
                         57  D5 00892           TSTL    CMD_KIND                                ; 1652
                         78  12 00894           BNEQ    129$                                    ;
```

```
                 04         55  D1 00896          CMPL    SUB_KIND, #4                                    ; 1653
                            73  12 00899          BNEQ    129$                                            ;
                 19     0C   AE  E9 0089B 125$:    BLBC    SYSTEM, 126$                                    ; 1656
                            03  DD 0089F          PUSHL   #3                                              ; 1658
00000000G        00         01  FB 008A1          CALLS   #1, DBG$GET_TEMPMEM                             ;
                 52         50  D0 008A8          MOVL    R0, ADVERB_NODE                                 ;
                       08   A2  D4 008AB          CLRL    8(ADVERB_NODE)                                  ;
                 68         52  D0 008AE          MOVL    ADVERB_NODE, (ADVERB_LINK)                      ;
                 58     08  A2  9E 008B1          MOVAB   8(ADVERB_NODE), ADVERB_LINK                     ;
                 62         0E  90 008B5          MOVB    #14, (ADVERB_NODE)                              ; 1660
                 19     08  AE  E9 008B8 126$:    BLBC    NOSYSTEM, 127$                                  ; 1663
                            03  DD 008BC          PUSHL   #3                                              ; 1665
00000000G        00         01  FB 008BE          CALLS   #1, DBG$GET_TEMPMEM                             ;
                 52         50  D0 008C5          MOVL    R0, ADVERB_NODE                                 ;
                       08   A2  D4 008C8          CLRL    8(ADVERB_NODE)                                  ;
                 68         52  D0 008CB          MOVL    ADVERB_NODE, (ADVERB_LINK)                      ;
                 58     08  A2  9E 008CE          MOVAB   8(ADVERB_NODE), ADVERB_LINK                     ;
                 62         0F  90 008D2          MOVB    #15, (ADVERB_NODE)                              ; 1667
                 19     04  AE  E9 008D5 127$:    BLBC    INTO, 128$                                      ; 1670
                            03  DD 008D9          PUSHL   #3                                              ; 1672
00000000G        00         01  FB 008DB          CALLS   #1, DBG$GET_TEMPMEM                             ;
                 52         50  D0 008E2          MOVL    R0, ADVERB_NODE                                 ;
                       08   A2  D4 008E5          CLRL    8(ADVERB_NODE)                                  ;
                 68         52  D0 008E8          MOVL    ADVERB_NODE, (ADVERB_LINK)                      ;
                 58     08  A2  9E 008EB          MOVAB   8(ADVERB_NODE), ADVERB_LINK                     ;
                 62         10  90 008EF          MOVB    #16, (ADVERB_NODE)                              ; 1674
                 19         6E  E9 008F2 128$:    BLBC    OVER, 129$                                      ; 1677
                            03  DD 008F5          PUSHL   #3                                              ; 1679
00000000G        00         01  FB 008F7          CALLS   #1, DBG$GET_TEMPMEM                             ;
                 52         50  D0 008FE          MOVL    R0, ADVERB_NODE                                 ;
                       08   A2  D4 00901          CLRL    8(ADVERB_NODE)                                  ;
                 68         52  D0 00904          MOVL    ADVERB_NODE, (ADVERB_LINK)                      ;
                 58     08  A2  9E 00907          MOVAB   8(ADVERB_NODE), ADVERB_LINK                     ;
                 62         11  90 0090B          MOVB    #17, (ADVERB_NODE)                              ; 1681
                 03         56  D1 0090E 129$:    CMPL    CMD_TYPE, #3                                    ; 1690
                            60  13 00911          BEQL    132$                                            ;
                            57  D5 00913          TSTL    CMD_KIND                                        ; 1691
                            5C  12 00915          BNEQ    132$                                            ;
                            04  DD 00917 130$:     PUSHL   #4                                              ; 1695
00000000G        00         01  FB 00919          CALLS   #1, DBG$GET_TEMPMEM                             ;
                 54         50  D0 00920          MOVL    R0, NOUN_NODE                                   ;
                       08   A4  D4 00923          CLRL    8(NOUN_NODE)                                    ;
                 69         54  D0 00926          MOVL    NOUN_NODE, (NOUN_LINK)                          ;
                 59     08  A4  9E 00929          MOVAB   8(NOUN_NODE), NOUN_LINK                         ;
00000000'        EF         01  90 0092D          MOVB    #1, DBG$GB_SET_BREAK_FLAG                       ; 1709
                       0C   AC  DD 00934          PUSHL   MESSAGE_VECT                                    ; 1713
                            09  DD 00937          PUSHL   #9                                              ; 1711
                 7E 00000000G   00  9A 00939      MOVZBL  DBG$GB_RADIX, -(SP)                             ; 1712
                            18  BB 00940          PUSHR   #^M<R3,R4>                                      ; 1711
00000000G        00         05  FB 00942          CALLS   #5, DBG$NPARSE_ADDRESS                          ;
                 55         50  D0 00949          MOVL    R0, STATUS                                      ;
                       00000000' EF  94 0094C     CLRB    DBG$GB_SET_BREAK_FLAG                           ; 1714
                 01         55  D1 00952          CMPL    STATUS, #1                                      ; 1720
                            1C  13 00955          BEQL    132$                                            ;
                            55  D5 00957          TSTL    STATUS                                          ; 1728
                            04  13 00959          BEQL    131$                                            ;
                 50         55  D0 0095B          MOVL    STATUS, R0                                      ; 1730
```

```
                                 04 0095E                RET
                                 01   DD 0095F 131$:     PUSHL   #1                                                  ;  1733
                      00000000'  EF   9F 00961           PUSHAB  DBG$CS_COMMA                                       ;
                                 53   DD 00967           PUSHL   R3                                                 ;
00000000G               00       03   FB 00969           CALLS   #3, DBG$NMATCH                                     ;
                        A4       50   E8 00970           BLBS    R0, 130$                                           ;
                                 01   DD 00973 132$:     PUSHL   #1                                                 ;  1742
                      00000000'  EF   9F 00975           PUSHAB  DBG$CS_CR                                          ;
                                 53   DD 0097B           PUSHL   R3                                                 ;
00000000G               00       03   FB 0097D           CALLS   #3, DBG$NMATCH                                     ;
                        33       50   E8 00984           BLBS    R0, 134$                                           ;
                        03       56   D1 00987           CMPL    CMD_TYPE, #3                                       ;  1749
                                 37   12 0098A           BNEQ    136$                                               ;
                        03   1C  AE   E8 0098C           BLBS    28(SP), 133$                                       ;  1770
                           00A5  31   00990               BRW     138$                                               ;
                                 03   DD 00993 133$:     PUSHL   #3                                                 ;  1753
00000000G               00       01   FB 00995           CALLS   #1, DBG$GET_TEMPMEM                                ;
                        52       50   D0 0099C           MOVL    R0, ADVERB_NODE                                    ;
                              08 A2   D4 0099F           CLRL    8(ADVERB_NODE)                                     ;
                        68       52   D0 009A2           MOVL    ADVERB_NODE, (ADVERB_LINK)                         ;
                        58   08  A2   9E 009A5           MOVAB   8(ADVERB_NODE), ADVERB_LINK                        ;
                                 62   94 009A9           CLRB    (ADVERB_NODE)                                      ;  1766
                              0C AC   DD 009AB           PUSHL   MESSAGE_VECT                                       ;  1770
                              04 A2   9F 009AE           PUSHAB  4(ADVERB_NODE)                                     ;  1769
                                 53   DD 009B1           PUSHL   R3                                                 ;
00000000G               00       03   FB 009B3           CALLS   #3, DBG$NSAVE_DECIMAL_INTEGER                      ;
                        03       50   E9 009BA 134$:     BLBC    R0, 135$                                           ;
                           00D6  31   009BD               BRW     142$                                               ;
                           00ED  31   009C0 135$:         BRW     145$                                               ;  1775
                                 01   DD 009C3 136$:     PUSHL   #1                                                 ;  1786
                      00000000'  EF   9F 009C5           PUSHAB  DBG$CS_WHEN                                        ;
                                 53   DD 009CB           PUSHL   R3                                                 ;
00000000G               00       03   FB 009CD           CALLS   #3, DBG$NMATCH                                     ;
                        39       50   E9 009D4           BLBC    R0, 137$                                           ;
                                 01   DD 009D7           PUSHL   #1                                                 ;  1793
                      00000000'  EF   9F 009D9           PUSHAB  DBG$CS_L_PAREN                                     ;
                                 53   DD 009DF           PUSHL   R3                                                 ;
00000000G               00       03   FB 009E1           CALLS   #3, DBG$NMATCH                                     ;
                        4D       50   E9 009E8           BLBC    R0, 138$                                           ;
                                 03   DD 009EB           PUSHL   #3                                                 ;
00000000G               00       01   FB 009ED           CALLS   #1, DBG$GET_TEMPMEM                                ;
                        52       50   D0 009F4           MOVL    R0, ADVERB_NODE                                    ;
                              08 A2   D4 009F7           CLRL    8(ADVERB_NODE)                                     ;
                        68       52   D0 009FA           MOVL    ADVERB_NODE, (ADVERB_LINK)                         ;
                        58   08  A2   9E 009FD           MOVAB   8(ADVERB_NODE), ADVERB_LINK                        ;
                        62       04   90 00A01           MOVB    #4, (ADVERB_NODE)                                  ;  1803
                              04 A2   9F 00A04           PUSHAB  4(ADVERB_NODE)                                     ;  1804
                                 53   DD 00A07           PUSHL   R3                                                 ;
00000000G               00       02   FB 00A09           CALLS   #2, DBG$NSAVE_BREAK_BUFFER                         ;
                                 01   DD 00A10 137$:     PUSHL   #1                                                 ;  1810
                      00000000'  EF   9F 00A12           PUSHAB  DBG$CS_DO                                          ;
                                 53   DD 00A18           PUSHL   R3                                                 ;
00000000G               00       03   FB 00A1A           CALLS   #3, DBG$NMATCH                                     ;
                        5E       50   E9 00A21           BLBC    R0, 141$                                           ;
                                 01   DD 00A24           PUSHL   #1                                                 ;  1817
                      00000000'  EF   9F 00A26           PUSHAB  DBG$CS_L_PAREN                                     ;
                                 53   DD 00A2C           PUSHL   R3                                                 ;
```

```
         00000000G  00          03  FB 00A2E        CALLS   #3, DBG$NMATCH                         ;
                    23          50  E8 00A35        BLBS    R0, 140$                               ;
                                01  DD 00A38 138$:  PUSHL   #1                                     ;
                       00000000' EF  9F 00A3A        PUSHAB  DBG$CS_CR                              ;
                                53  DD 00A40        PUSHL   R3                                     ;
         00000000G  00          03  FB 00A42        CALLS   #3, DBG$NMATCH                         ;
                    4E          50  E9 00A49        BLBC    R0, 143$                               ;
                       000280D0 8F  DD 00A4C 139$:  PUSHL   #164048                                ;
         00000000G  00          01  FB 00A52        CALLS   #1, DBG$NMAKE_ARG_VECT                 ;
                                51  11 00A59        BRB     144$                                   ;
                                03  DD 00A5B 140$:  PUSHL   #3                                     ;
         00000000G  00          01  FB 00A5D        CALLS   #1, DBG$GET_TEMPMEM                    ;
                    52          50  D0 00A64        MOVL    R0, ADVERB_NODE                        ;
                             08 A2  D4 00A67        CLRL    8(ADVERB_NODE)                         ;
                    68          52  D0 00A6A        MOVL    ADVERB_NODE, (ADVERB_LINK)             ;
                    58       08 A2  9E 00A6D        MOVAB   8(ADVERB_NODE), ADVERB_LINK            ;
                    62          05  90 00A71        MOVB    #5, (ADVERB_NODE)                      ; 1827
                             04 A2  9F 00A74        PUSHAB  4(ADVERB_NODE)                         ; 1828
                                53  DD 00A77        PUSHL   R3                                     ;
         00000000G  00          02  FB 00A79        CALLS   #2, DBG$NSAVE_BREAK_BUFFER             ;
                                14  11 00A80        BRB     142$                                   ; 1829
                                01  DD 00A82 141$:  PUSHL   #1                                     ; 1836
                       00000000' EF  9F 00A84        PUSHAB  DBG$CS_CR                              ;
                                53  DD 00A8A        PUSHL   R3                                     ;
         00000000G  00          03  FB 00A8C        CALLS   #3, DBG$NMATCH                         ;
                    04          50  E9 00A93        BLBC    R0, 143$                               ;
                    50          01  D0 00A96 142$:  MOVL    #1, R0                                 ;
                                    04 00A99        RET                                            ;
                                53  DD 00A9A 143$:  PUSHL   R3                                     ; 1837
         00000000G  00          01  FB 00A9C        CALLS   #1, DBG$NNEXT_WORD                     ;
                                50  DD 00AA3        PUSHL   R0                                     ;
         00000000G  00          01  FB 00AA5        CALLS   #1, DBG$NSYNTAX_ERROR                  ;
                0C  BC          50  D0 00AAC 144$:  MOVL    R0, @MESSAGE_VECT                      ;
                    50          04  D0 00AB0 145$:  MOVL    #4, R0                                 ; 1838
                                    04 00AB3        RET                                            ; 1839

; Routine Size:  2740 bytes,    Routine Base:  DBG$CODE + 0037
```

```
;  1715          1840  1 GLOBAL ROUTINE DBG$EVENT_SEMANTICS(VERB_NODE, MESSAGE_VECT) =
;  1716          1841  1 !
;  1717          1842  1 !  FUNCTION
;  1718          1843  1 !       This routine builds the event data stuctures based on the
;  1719          1844  1 !       parse tree passed via the VERB_NODE address.  This routine
;  1720          1845  1 !       is called from DBGNSET.
;  1721          1846  1 !
;  1722          1847  1 !  INPUTS
;  1723          1848  1 !       VERB_NODE:      A pointer to a DEBUG verb node, which defines,
;  1724          1849  1 !                       along with the verb's descendants, the desired
;  1725          1850  1 !                       command.
;  1726          1851  1 !       MESSAGE_VECT:   The address of a longword to contain the address
;  1727          1852  1 !                       of a message argument vector.
;  1728          1853  1 !
;  1729          1854  1 !  OUTPUTS
;  1730          1855  1 !       The event data structure(s) are updated to include the new event,
;  1731          1856  1 !       and may include:
;  1732          1857  1 !
;  1733          1858  1 !           EVENT$CMD_QUEUE,
;  1734          1859  1 !           EVENT$WHEN_QUEUE,
;  1735          1860  1 !           EVENT$DO_LIST_QUEUE,
;  1736          1861  1 !           EVENT$PAGE_QUEUE
;  1737          1862  1 !
;  1738          1863  1 !       STS$K_SEVERE is return on error, otherwise STS$K_SUCCESS
;  1739          1864  1 !
;  1740          1865  2     BEGIN
;  1741          1866  2
;  1742          1867  2     MAP
;  1743          1868  2         VERB_NODE :     REF DBG$VERB_NODE;      ! Verb node pointer
;  1744          1869  2
;  1745          1870  2     LOCAL
;  1746          1871  2         DUMMY,
;  1747          1872  2         EVENT_QUEUE :   QUEUE_HEAD,             ! Working queue head
;  1748          1873  2         QUEUE_ENTRY :   REF EVENT$EVENT_DESCRIPTOR, ! Event queue entry pointer
;  1749          1874  2         NEXTQ_ENTRY :   REF EVENT$EVENT_DESCRIPTOR, ! Event queue entry pointer
;  1750          1875  2         EVENT_ENTRY :   REF EVENT$EVENT_DESCRIPTOR, ! Event entry pointer
;  1751          1876  2         DO_ENTRY :      REF EVENT$DO_LIST_DESCRIPTOR, ! DO List entry pointer
;  1752          1877  2         WHEN_ENTRY :    REF EVENT$WHEN_DESCRIPTOR, ! WHEN entry pointer
;  1753          1878  2         ADVERB_NODE :   REF DBG$ADVERB_NODE,    ! Adverb node link
;  1754          1879  2         NOUN_NODE :     REF DBG$NOUN_NODE,      ! Noun node link
;  1755          1880  2
;  1756          1881  2         PRIMARY :       REF DBG$PRIMARY,        ! Event primary descriptor
;  1757          1882  2         VALUE :         REF DBG$VALDESC,        ! Event value descriptor
;  1758          1883  2
;  1759          1884  2         ADDRESS :       VECTOR [2],             ! Event address
;  1760          1885  2         ADR_TYP,                                ! Event address type
;  1761          1886  2         LENGTH,                                 ! Event value length
;  1762          1887  2
;  1763          1888  2         STRING :        REF VECTOR [, BYTE],    ! Text pointer
;  1764          1889  2         COUNTER;                                ! Counter variable
;  1765          1890  2
;  1766          1891  2
;  1767          1892  2     ! Initialize the working queue to be empty.
;  1768          1893  2     !
;  1769          1894  2     EVENT_QUEUE [L_QUEUE_FLINK] = EVENT_QUEUE;
;  1770          1895  2     EVENT_QUEUE [L_QUEUE_BLINK] = EVENT_QUEUE;
;  1771          1896  2
```

```
;  1772          1897  2
;  1773          1898  2        ! Get an event descriptor from permanent memory, and
;  1774          1899  2        ! link it into the working queue. Also inititialize the
;  1775          1900  2        ! inside queue header.
;  1776          1901  2        !
;  1777          1902  2        EVENT_ENTRY = DBG$GET_MEMORY (EVENT$K_EVENT_DESCRIPTOR_SIZE);
;  1778          1903  2        INSQUE (.EVENT_ENTRY, EVENT_QUEUE);
;  1779          1904  2        EVENT_ENTRY[EVENT$L_EXC_FLINK] = .EVENT_ENTRY;
;  1780          1905  2        EVENT_ENTRY[EVENT$L_EXC_BLINK] = .EVENT_ENTRY;
;  1781          1906  2
;  1782          1907  2
;  1783          1908  2        ! Set the event's command type, found in the verb.
;  1784          1909  2        !
;  1785          1910  2        SELECTONE .VERB_NODE [DBG$B_VERB_COMPOSITE] OF
;  1786          1911  2            SET
;  1787          1912  2
;  1788          1913  2            [EVENT$K_SET_BREAK]:
;  1789          1914  2                EVENT_ENTRY [EVENT$B_CMD_TYPE] = EVENT$K_TYPE_BREAK;
;  1790          1915  2
;  1791          1916  2            [EVENT$K_SET_BREAK_EXC]:
;  1792          1917  3                BEGIN
;  1793          1918  3                EVENT_ENTRY [EVENT$B_CMD_TYPE] = EVENT$K_TYPE_BREAK;
;  1794          1919  3                EVENT_ENTRY [EVENT$B_CMD_KIND] = EVENT$K_KIND_EXC;
;  1795          1920  3                END;
;  1796          1921  2
;  1797          1922  2            [EVENT$K_SET_TRACE]:
;  1798          1923  2                EVENT_ENTRY [EVENT$B_CMD_TYPE] = EVENT$K_TYPE_TRACE;
;  1799          1924  2
;  1800          1925  2            [EVENT$K_SET_WATCH]:
;  1801          1926  3                BEGIN
;  1802          1927  3                EVENT_ENTRY [EVENT$B_CMD_TYPE] = EVENT$K_TYPE_WATCH;
;  1803          1928  3                EVENT_ENTRY [EVENT$B_CMD_KIND] = EVENT$K_KIND_ACC;
;  1804          1929  3                EVENT_ENTRY [EVENT$B_SUB_KIND] = EVENT$K_ACC_MODFY
;  1805          1930  3                END;
;  1806          1931  2
;  1807          1932  2            [EVENT$K_SET_STEP,
;  1808          1933  2             EVENT$K_STEP]:
;  1809          1934  2                EVENT_ENTRY [EVENT$B_CMD_TYPE] = EVENT$K_TYPE_STEPS;
;  1810          1935  2
;  1811          1936  2            [OTHERWISE]:
;  1812          1937  2                $DBG_ERROR('DBGEVENT\EVENT_SEMANTICS 10');
;  1813          1938  2
;  1814          1939  2            TES;
;  1815          1940  2
;  1816          1941  2
;  1817          1942  2        ! Initialize the Deleted, Silent, Once (Temporary), Threaded, and
;  1818          1943  2        ! Has Value Descriptor flags.
;  1819          1944  2        !
;  1820          1945  2        EVENT_ENTRY [EVENT$B_CMD_FLAGS] = 0;
;  1821          1946  2
;  1822          1947  2
;  1823          1948  2        ! Initialize the step-type flags.  If the command is STEP or SET STEP,
;  1824          1949  2        ! we set up the Event Entry using the current default stepping modes.
;  1825          1950  2        !
;  1826          1951  2        EVENT_ENTRY[EVENT$L_STEP_FLAGS] = 0;
;  1827          1952  2        IF .EVENT_ENTRY [EVENT$B_CMD_TYPE] EQLU EVENT$K_TYPE_STEPS
;  1828          1953  2        THEN
```

```
;  1829          1954  3          BEGIN
;  1830          1955  3          EVENT_ENTRY[EVENT$V_STEP_LINE] = .DBG$GB_STP_PTR[EVENT$V_STEPPING_LINE];
;  1831          1956  3          EVENT_ENTRY[EVENT$V_STEP_OVER] = .DBG$GB_STP_PTR[EVENT$V_STEPPING_OVER];
;  1832          1957  3          EVENT_ENTRY[EVENT$V_STEP_SOURCE] =
;  1833          1958  3                              .DBG$GB_STP_PTR[EVENT$V_STEPPING_SOURCE];
;  1834          1959  3          EVENT_ENTRY[EVENT$V_OVRD_SOURCE] = FALSE;
;  1835          1960  3          EVENT_ENTRY[EVENT$V_STEP_NOSYSTEM] =
;  1836          1961  3                              .DBG$GB_STP_PTR [EVENT$V_STEPPING_NOSYSTEM];
;  1837          1962  3          EVENT_ENTRY[EVENT$V_STEP_NOSILENT] =
;  1838          1963  3                             .DBG$GB_STP_PTR [EVENT$V_STEPPING_NOSILENT];
;  1839          1964  3          END
;  1840          1965
;  1841          1966
;  1842          1967          ! If the command is SET BREAK, SET TRACE, or SET WATCH, we initialize the
;  1843          1968          ! various flags to FALSE.
;  1844          1969          !
;  1845          1970          ELSE
;  1846          1971              BEGIN
;  1847          1972              EVENT_ENTRY[EVENT$V_STEP_LINE] = FALSE;
;  1848          1973              EVENT_ENTRY[EVENT$V_STEP_OVER] = FALSE;
;  1849          1974              EVENT_ENTRY[EVENT$V_STEP_SOURCE] = FALSE;
;  1850          1975              EVENT_ENTRY[EVENT$V_OVRD_SOURCE] = FALSE;
;  1851          1976              EVENT_ENTRY[EVENT$V_STEP_NOSYSTEM] = FALSE;
;  1852          1977              EVENT_ENTRY[EVENT$V_STEP_NOSILENT] = FALSE;
;  1853          1978              END;
;  1854          1979
;  1855          1980
;  1856          1981          ! Initialize the Primary pointer to NULL.
;  1857          1982          !
;  1858          1983          EVENT_ENTRY [EVENT$L_PRIMARY] = 0;
;  1859          1984
;  1860          1985
;  1861          1986  2       ! Initialize the event entry /AFTER count to 1.
;  1862          1987  2       !
;  1863          1988  2       EVENT_ENTRY [EVENT$L_AFTER_COUNT] = 1;
;  1864          1989  2
;  1865          1990  2
;  1866          1991  2       ! Initialize the WHEN and DO List pointers to NULL.
;  1867          1992  2       !
;  1868          1993  2       EVENT_ENTRY [EVENT$L_WHEN] = 0;
;  1869          1994  2       EVENT_ENTRY [EVENT$L_DO] = 0;
;  1870          1995  2
;  1871          1996  2
;  1872          1997  2       ! Point to the verb's noun list.
;  1873          1998  2       !
;  1874          1999  2       NOUN_NODE = .VERB_NODE [DBG$L_VERB_OBJECT_PTR];
;  1875          2000  2
;  1876          2001  2
;  1877          2002  2       ! Point to the verb's adverb list.
;  1878          2003  2       !
;  1879          2004  2       ADVERB_NODE = .VERB_NODE [DBG$L_VERB_ADVERB_PTR];
;  1880          2005  2
;  1881          2006  2
;  1882          2007  2       ! For each existing adverb, process it.
;  1883          2008  2       !
;  1884          2009  2       WHILE .ADVERB_NODE NEQA 0 DO
;  1885          2010  3           BEGIN
```

```
;  1886          2011  3
;  1887          2012
;  1888          2013              ! Process adverbs based on the literal set by the
;  1889          2014              ! syntactic processing.
;  1890          2015              !
;  1891          2016              SELECTONE .ADVERB_NODE [DBG$B_ADVERB_LITERAL] OF
;  1892          2017                  SET
;  1893          2018
;  1894          2019
;  1895          2020                  ! Process the /AFTER : <n> adverb.
;  1896          2021                  !
;  1897          2022                  [ADVERB_AFTER]:
;  1898          2023                      EVENT_ENTRY [EVENT$L_AFTER_COUNT] =
;  1899          2024                          .ADVERB_NODE [DBG$L_ADVERB_VALUE];
;  1900          2025
;  1901          2026
;  1902          2027                  ! Process the /TEMPORARY adverb.
;  1903          2028                  !
;  1904          2029                  [ADVERB_TEMPORARY]:
;  1905          2030                      EVENT_ENTRY [EVENT$V_ONCE_ONLY] = TRUE;
;  1906          2031
;  1907          2032
;  1908          2033                  ! Process the /SILENT adverb.
;  1909          2034                  !
;  1910          2035                  [ADVERB_SILENT]:
;  1911          2036  4                   BEGIN
;  1912          2037  4                   EVENT_ENTRY [EVENT$V_SILENT] = TRUE;
;  1913          2038  4                   EVENT_ENTRY [EVENT$V_STEP_NOSILENT] = FALSE;
;  1914          2039  4                   EVENT_ENTRY [EVENT$V_OVRD_NOSILENT] = TRUE;
;  1915          2040  4                   END;
;  1916          2041
;  1917          2042
;  1918          2043              ! Process the /NOSILENT adverb.
;  1919          2044              !
;  1920          2045              [ADVERB_NOSILENT]:
;  1921          2046  4                   BEGIN
;  1922          2047  4                   EVENT_ENTRY [EVENT$V_SILENT] = FALSE;
;  1923          2048  4                   EVENT_ENTRY [EVENT$V_STEP_NOSILENT] = TRUE;
;  1924          2049  4                   EVENT_ENTRY [EVENT$V_OVRD_NOSILENT] = TRUE;
;  1925          2050  4                   END;
;  1926          2051
;  1927          2052
;  1928          2053              ! Process the WHEN adverb.
;  1929          2054              !
;  1930          2055              [ADVERB_WHEN]:
;  1931          2056  4                   BEGIN
;  1932          2057  4                   LOCAL
;  1933          2058  4                       LENGTH;
;  1934          2059  4
;  1935          2060  4                   ! Get a WHEN descriptor from permanent memory, and
;  1936          2061  4                   ! link it into the WHEN queue.
;  1937          2062  4                   !
;  1938          2063  4                   WHEN_ENTRY = DBG$GET_MEMORY (EVENT$K_WHEN_DESCRIPTOR_SIZE);
;  1939          2064  4                   INSQUE (.WHEN_ENTRY, EVENT$WHEN_QUEUE);
;  1940          2065  4
;  1941          2066  4
;  1942          2067  4                   ! Point the event entry to the WHEN entry.
```

```
;  1943          2068  4                    !
;  1944          2069  4                    EVENT_ENTRY [EVENT$L_WHEN] = .WHEN_ENTRY;
;  1945          2070  4
;  1946          2071  4
;  1947          2072  4                    ! Put a <cr> at the end of the WHEN string.
;  1948          2073  4                    !
;  1949          2074  4                    STRING = .ADVERB_NODE [DBG$L_ADVERB_VALUE];
;  1950          2075  4                    LENGTH = .(.STRING)<0,16,0>;
;  1951          2076  4                    STRING [.LENGTH+1] = DBG$K_CAR_RETURN;
;  1952          2077  4
;  1953          2078  4
;  1954          2079  4                    ! Fill in the WHEN entry.
;  1955          2080  4                    !
;  1956          2081  4                    WHEN_ENTRY [EVENT$L_WHEN_COUNT] = 1;
;  1957          2082  4                    WHEN_ENTRY [EVENT$L_WHEN_POINT] = .STRING;
;  1958          2083  3                    END;
;  1959          2084  3
;  1960          2085  3
;  1961          2086  3                ! Process the DO adverb.
;  1962          2087  3                !
;  1963          2088  3                [ADVERB_DO]:
;  1964          2089  4                    BEGIN
;  1965          2090  4                    LOCAL
;  1966          2091  4                        LENGTH;
;  1967          2092  4
;  1968          2093  4
;  1969          2094  4                    ! Get a DO LIST descriptor from permanent memory, and
;  1970          2095  4                    ! link it into the DO LIST queue.
;  1971          2096  4                    !
;  1972          2097  4                    DO_ENTRY = DBG$GET_MEMORY (EVENT$K_DO_LIST_DESCRIPTOR_SIZE);
;  1973          2098  4                    INSQUE (.DO_ENTRY, EVENT$DO_LIST_QUEUE);
;  1974          2099  4
;  1975          2100  4
;  1976          2101  4                    ! Point the event entry to the DO LIST entry.
;  1977          2102  4                    !
;  1978          2103  4                    EVENT_ENTRY [EVENT$L_DO] = .DO_ENTRY;
;  1979          2104  4
;  1980          2105  4
;  1981          2106  4                    ! Put a <cr> at the end of the DO string.
;  1982          2107  4                    !
;  1983          2108  4                    STRING = .ADVERB_NODE [DBG$L_ADVERB_VALUE];
;  1984          2109  4                    LENGTH = .(.STRING)<0,16,0>;
;  1985          2110  4                    STRING [.LENGTH+1] = DBG$K_CAR_RETURN;
;  1986          2111  4
;  1987          2112  4
;  1988          2113  4                    ! Fill in the DO entry.
;  1989          2114  4                    !
;  1990          2115  4                    DO_ENTRY [EVENT$L_DO_LIST_COUNT] = 1;
;  1991          2116  4                    DO_ENTRY [EVENT$L_DO_LIST_POINT] = .STRING;
;  1992          2117  3                    END;
;  1993          2118  3
;  1994          2119  3
;  1995          2120  3                ! Process the /READ, /WRITE, /MODIFY, /EXECUTE, and /RETURN
;  1996          2121  3                ! access adverbs.
;  1997          2122  3                !
;  1998          2123  3                [ADVERB_READ,ADVERB_WRITE,ADVERB_MODIFY,
;  1999          2124  3                 ADVERB_EXECUTE,ADVERB_RETURN]:
```

```
;  2000          2125  4                BEGIN
;  2001          2126  4
;  2002          2127  4
;  2003          2128  4                ! Define the command kind to be access.
;  2004          2129  4                !
;  2005          2130  4                EVENT_ENTRY [EVENT$B_CMD_KIND] = EVENT$K_KIND_ACC;
;  2006          2131  4
;  2007          2132  4
;  2008          2133  4                ! Define the sub-kind as appropriate.
;  2009          2134  4                !
;  2010          2135  4                EVENT_ENTRY [EVENT$B_SUB_KIND] =
;  2011          2136  5                    (SELECTONE .ADVERB_NODE [DBG$B_ADVERB_LITERAL] OF
;  2012          2137  5                               SET
;  2013          2138  5                               [ADVERB_READ]:
;  2014          2139  5                                   EVENT$K_ACC_READ;
;  2015          2140  5                               [ADVERB_WRITE]:
;  2016          2141  5                                   EVENT$K_ACC_WRIT;
;  2017          2142  5                               [ADVERB_MODIFY]:
;  2018          2143  5                                   EVENT$K_ACC_MDFY;
;  2019          2144  5                               [ADVERB_EXECUTE]:
;  2020          2145  5                                   EVENT$K_ACC_EXEC;
;  2021          2146  5                               [ADVERB_RETURN]:
;  2022          2147  5                                   EVENT$K_ACC_RTRN;
;  2023          2148  5                               [OTHERWISE]:
;  2024          2149  5                                   $DBG_ERROR('DBGEVENT\EVENT_SEMANTICS 20');
;  2025          2150  5                               TES
;  2026          2151  4                    );
;  2027          2152  3                END;
;  2028          2153  3
;  2029          2154  3
;  2030          2155  3            ! Process the /EXCEPTION adverb.
;  2031          2156  3            !
;  2032          2157  3            [ADVERB_EXCEPTION]:
;  2033          2158  4                BEGIN
;  2034          2159  4                EVENT_ENTRY [EVENT$B_CMD_KIND] = EVENT$K_KIND_EXC;
;  2035          2160  4                EVENT_ENTRY [EVENT$B_SUB_KIND] = EVENT$K_EXC_EXC;
;  2036          2161  3                END;
;  2037          2162  3
;  2038          2163  3
;  2039          2164  3            ! Process the /CALL, /RETURN, /BRANCH, /OPCODE, /INSTRUCTION and
;  2040          2165  3            ! /LINE instruction-kind adverbs.
;  2041          2166  3            !
;  2042          2167  3            [ADVERB_CALL,
;  2043          2168  3             ADVERB_RETURN,
;  2044          2169  3             ADVERB_BRANCH,
;  2045          2170  3             ADVERB_OPCODE,
;  2046          2171  3             ADVERB_INSTRUCTION,
;  2047          2172  3             ADVERB_LINE]:
;  2048          2173  4                BEGIN
;  2049          2174  4                LOCAL
;  2050          2175  4                    LINENO,      ! Line number returned from PC_TO_LINE
;  2051          2176  4                    MODRST,      ! Module RST returned from PC_TO_LINE
;  2052          2177  4                    STATUS,      ! Status returned from PC_TO_LINE
;  2053          2178  4                    STMTNO;      ! Statement number returned from PC_TO_LINE
;  2054          2179  4
;  2055          2180  4                ! Define the event kind to be instruction.
;  2056          2181  4                !
```

```
;  2057          2182  4                EVENT_ENTRY [EVENT$B_CMD_KIND] = EVENT$K_KIND_INS;
;  2058          2183  4
;  2059          2184  4
;  2060          2185  4                ! Define the subkind according to the adverb.
;  2061          2186  4                !
;  2062          2187  4                EVENT_ENTRY [EVENT$B_SUB_KIND] =
;  2063          2188  5                    (SELECTONE .ADVERB_NODE [DBG$B_ADVERB_LITERAL] OF
;  2064          2189  5                        SET
;  2065          2190  5                        [ADVERB_CALL]:
;  2066          2191  5                            EVENT$K_INS_CALL;
;  2067          2192  5                        [ADVERB_BRANCH]:
;  2068          2193  5                            EVENT$K_INS_BRAN;
;  2069          2194  5                        [ADVERB_LINE]:
;  2070          2195  5                            EVENT$K_INS_LINE;
;  2071          2196  5                        [ADVERB_OPCODE]:
;  2072          2197  5                            EVENT$K_INS_USER;
;  2073          2198  5                        [ADVERB_INSTRUCTION]:
;  2074          2199  5                            EVENT$K_INS_EVRY;
;  2075          2200  5                        [OTHERWISE]:
;  2076          2201  5                            $DBG_ERROR('DBGEVENT\EVENT_SEMANTICS 30');
;  2077          2202  5                        TES
;  2078          2203  4                    );
;  2079          2204  4
;  2080          2205  4                ! Set the line stepping flag appropriately.
;  2081          2206  4                !
;  2082          2207  4                EVENT_ENTRY [EVENT$V_STEP_LINE] =
;  2083          2208  4                    (.EVENT_ENTRY [EVENT$B_SUB_KIND] EQLU EVENT$K_INS_LINE);
;  2084          2209  4
;  2085          2210  4
;  2086          2211  4                ! We're not in the middle of a step over....
;  2087          2212  4                !
;  2088          2213  4                EVENT_ENTRY [EVENT$V_STEP_BPT] = FALSE;
;  2089          2214  4
;  2090          2215  4
;  2091          2216  4                ! Initialize the LO and HI PC values by calling PC_TO_LINE
;  2092          2217  4                ! with the current PC. This gives us a PC range for the
;  2093          2218  4                ! line number containing the current PC. This PC range
;  2094          2219  4                ! will be used later to determine when a step by line ends.
;  2095          2220  4                ! If PC_TO_LINE returns FALSE we are not in a line. In that
;  2096          2221  4                ! case, initialize LO and HI PC to 1 and 0, so we will
;  2097          2222  4                ! immediately be out of the "range" the next time we check.
;  2098          2223  4                !
;  2099          2224  4                STATUS = 0;
;  2100          2225  4                MODRST = 0;
;  2101          2226  4                IF .EVENT_ENTRY [EVENT$B_SUB_KIND] EQLU EVENT$K_INS_LINE
;  2102          2227  4                THEN
;  2103          2228  4                    STATUS = DBG$PC_TO_LINE_LOOKUP(
;  2104          2229  4                                .DBG$RONFRAME[DBG$L_USER_PC],
;  2105          2230  4                                LINENO,
;  2106          2231  4                                STMTNO,
;  2107          2232  4                                EVENT_ENTRY[EVENT$L_STEP_LO_PC],
;  2108          2233  4                                EVENT_ENTRY[EVENT$L_STEP_HI_PC],
;  2109          2234  4                                MODRST);
;  2110          2235  4                IF NOT .STATUS
;  2111          2236  4                THEN
;  2112          2237  4                    BEGIN
;  2113          2238  5
```

```
;  2114          2239  5                 EVENT_ENTRY[EVENT$L_STEP_LO_PC] = 1;
;  2115          2240  5                 EVENT_ENTRY[EVENT$L_STEP_HI_PC] = 0;
;  2116          2241                    END;
;  2117          2242                END;
;  2118          2243
;  2119          2244
;  2120          2245            ! Process the /SOURCE adverb.
;  2121          2246            !
;  2122          2247            [ADVERB_SOURCE]:
;  2123          2248                BEGIN
;  2124          2249                EVENT_ENTRY [EVENT$V_STEP_SOURCE] = TRUE;
;  2125          2250                EVENT_ENTRY [EVENT$V_OVRD_SOURCE] = TRUE;
;  2126          2251                END;
;  2127          2252
;  2128          2253
;  2129          2254            ! Process the /NOSOURCE adverb.
;  2130          2255            !
;  2131          2256            [ADVERB_NOSOURCE]:
;  2132          2257                BEGIN
;  2133          2258                EVENT_ENTRY [EVENT$V_STEP_SOURCE] = FALSE;
;  2134          2259                EVENT_ENTRY [EVENT$V_OVRD_SOURCE] = TRUE;
;  2135          2260                END;
;  2136          2261
;  2137          2262
;  2138          2263            ! Process the /SYSTEM adverb.
;  2139          2264            !
;  2140          2265            [ADVERB_SYSTEM]:
;  2141          2266                BEGIN
;  2142          2267                EVENT_ENTRY [EVENT$V_STEP_NOSYSTEM] = FALSE;
;  2143          2268                EVENT_ENTRY [EVENT$V_OVRD_NOSYSTEM] = TRUE;
;  2144          2269                END;
;  2145          2270
;  2146          2271
;  2147          2272            ! Process the /NOSYSTEM adverb.
;  2148          2273            !
;  2149          2274            [ADVERB_NOSYSTEM]:
;  2150          2275  4             BEGIN
;  2151          2276  4             EVENT_ENTRY [EVENT$V_STEP_NOSYSTEM] = TRUE;
;  2152          2277  4             EVENT_ENTRY [EVENT$V_OVRD_NOSYSTEM] = TRUE;
;  2153          2278                END;
;  2154          2279
;  2155          2280
;  2156          2281            ! Process the /INTO adverb.
;  2157          2282            !
;  2158          2283            [ADVERB_INTO]:
;  2159          2284  4             BEGIN
;  2160          2285  4             EVENT_ENTRY [EVENT$V_STEP_OVER] = FALSE;
;  2161          2286  4             EVENT_ENTRY [EVENT$V_OVRD_OVER] = TRUE;
;  2162          2287                END;
;  2163          2288
;  2164          2289
;  2165          2290            ! Process the /OVER adverb.
;  2166          2291            !
;  2167          2292            [ADVERB_OVER]:
;  2168          2293  4             BEGIN
;  2169          2294  4             EVENT_ENTRY [EVENT$V_STEP_OVER] = TRUE;
;  2170          2295  4             EVENT_ENTRY [EVENT$V_OVRD_OVER] = TRUE;
```

```
;  2171          2296  3                END;
;  2172          2297  3
;  2173          2298  3
;  2174          2299  3                ! An invalid adverb literal - yell !
;  2175          2300  3                !
;  2176          2301  3                [OTHERWISE]:
;  2177          2302  3                    $DBG_ERROR('DBGEVENT\EVENT_SEMANTICS 40');
;  2178          2303  3
;  2179          2304  3                TES;
;  2180          2305  3
;  2181          2306  3
;  2182          2307  3            ! Point to the next adverb in the list.
;  2183          2308  3            !
;  2184          2309  3            ADVERB_NODE = .ADVERB_NODE [DBG$L_ADVERB_LINK];
;  2185          2310  3            END;
;  2186          2311  2
;  2187          2312  2
;  2188          2313  2        ! Now, based on the command kind (access or instruction),
;  2189          2314  2        ! process the noun node list.
;  2190          2315  2        !
;  2191          2316  2        SELECTONE .EVENT_ENTRY [EVENT$B_CMD_KIND] OF
;  2192          2317  2            SET
;  2193          2318  2
;  2194          2319  2
;  2195          2320  2            ! Access events....
;  2196          2321  2            !
;  2197          2322  2            [EVENT$K_KIND_ACC]:
;  2198          2323  2
;  2199          2324  2                IF .EVENT_ENTRY [EVENT$B_CMD_TYPE] NEQU EVENT$K_TYPE_STEPS
;  2200          2325  2                THEN
;  2201          2326  3                    BEGIN
;  2202          2327  3
;  2203          2328  3                    LOCAL
;  2204          2329  3                        SYMID_LIST; ! Pointer to SYMID
;  2205          2330  3
;  2206          2331  3
;  2207          2332  3                    ! Make sure there's a noun node present.
;  2208          2333  3                    !
;  2209          2334  3                    IF .NOUN_NODE EQLA 0
;  2210          2335  3                    THEN
;  2211          2336  3                        $DBG_ERROR('DBGEVENT\EVENT_SEMANTICS 50');
;  2212          2337  3
;  2213          2338  3
;  2214          2339  3                    ! Build duplicate event entries, one for each
;  2215          2340  3                    ! address found in the noun node list.
;  2216          2341  3                    !
;  2217          2342  3                    WHILE TRUE DO
;  2218          2343  4                        BEGIN
;  2219          2344  4
;  2220          2345  4
;  2221          2346  4                        ! Obtain the primary descriptor.
;  2222          2347  4                        !
;  2223          2348  4                        PRIMARY = .NOUN_NODE [DBG$L_NOUN_VALUE];
;  2224          2349  4
;  2225          2350  4
;  2226          2351  4                        ! Obtain the symid list, and lock it.
;  2227          2352  4                        !
```

```
;  2228          2353  4                    DBG$NGET_SYMID (.PRIMARY, SYMID_LIST, DUMMY);
;  2229          2354  4                    DBG$STA_LOCK_SYMID (.SYMID_LIST);
;  2230          2355  4
;  2231          2356  4
;  2232          2357  4                    ! Save the primary descriptor by making a copy of the primary
;  2233          2358  4                    ! descriptor out of permanent storage.
;  2234          2359  4                    !
;  2235          2360  4                    DBG$NCOPY_DESC (.PRIMARY, EVENT_ENTRY[EVENT$L_PRIMARY],
;  2236          2361  4                                                  DUMMY, TRUE);
;  2237          2362  4
;  2238          2363  4
;  2239          2364  4                    ! Now, get and save the VMS descriptor. (We do this
;  2240          2365  4                    ! after saving the Primary since PRIM_TO_VAL sometimes
;  2241          2366  4                    ! modifies the Primary.)
;  2242          2367  4                    !
;  2243          2368  4                    DBG$PRIM_TO_VAL (.PRIMARY, DBG$K_V_VALUE_DESC, VALUE);
;  2244          2369  4                    CH$MOVE (12, VALUE [DBG$A_VALUE_VMSDESC],
;  2245          2370  4                                 EVENT_ENTRY [EVENT$A_VMSDESC]);
;  2246          2371  4
;  2247          2372  4
;  2248          2373  4                    ! Next, we determine the address where we put the
;  2249          2374  4                    ! breakpoint. We pass our Primary into DBG$NGET_ADDRESS.
;  2250          2375  4                    ! For routines and entry points, NGET_ADDRESS takes a
;  2251          2376  4                    ! flag that says to return the "break address" instead
;  2252          2377  4                    ! of the start address. For routines, the break address
;  2253          2378  4                    ! is after the routine prolog (or, if there is no prolog,
;  2254          2379  4                    ! it is at routine_start+2).
;  2255          2380  4                    !
;  2256          2381  4                    EVENT_ENTRY [EVENT$V_BREAK_ADDRESS] = FALSE;
;  2257          2382  5                    IF ((.EVENT_ENTRY [EVENT$B_CMD_TYPE] EQL EVENT$K_TYPE_BREAK) OR
;  2258          2383  5                        (.EVENT_ENTRY [EVENT$B_CMD_TYPE] EQL EVENT$K_TYPE_TRACE))
;  2259          2384  4                                          AND
;  2260          2385  5                        (.EVENT_ENTRY [EVENT$B_SUB_KIND] NEQ EVENT$K_ACC_RTRN)
;  2261          2386  4                    THEN
;  2262          2387  5                        BEGIN
;  2263          2388  6                        IF  (.PRIMARY[DBG$B_DHDR_TYPE] EQL DBG$K_PRIMARY_DESC)
;  2264          2389  5                                                  AND
;  2265          2390  6                           ((.PRIMARY[DBG$B_DHDR_KIND] EQL RST$K_ROUTINE) OR
;  2266          2391  6                            (.PRIMARY[DBG$B_DHDR_KIND] EQL RST$K_ENTRY))
;  2267          2392  5                        THEN
;  2268          2393  5                            EVENT_ENTRY [EVENT$V_BREAK_ADDRESS] = TRUE;
;  2269          2394  4                        END;
;  2270          2395  4
;  2271          2396  4
;  2272          2397  4                    ! Get the address where we place the breakpoint.
;  2273          2398  4                    !
;  2274          2399  4                    IF NOT DBG$NGET_ADDRESS(.PRIMARY, ADDRESS[0], ADR_TYP,
;  2275          2400  4                            .EVENT_ENTRY[EVENT$V_BREAK_ADDRESS], .MESSAGE_VECT)
;  2276          2401  4                    THEN
;  2277          2402  4                        RETURN STS$K_SEVERE;
;  2278          2403  4                    EVENT_ENTRY [EVENT$L_ADDRESS] = .ADDRESS[0];
;  2279          2404  4
;  2280          2405  4
;  2281          2406  4                    ! We may still need to adjust the address of the breakpoint
;  2282          2407  4                    ! by +2 to get past the entry mask of a routine. This
;  2283          2408  4                    ! will be the case if the user set a break on an absolute
;  2284          2409  4                    ! address which happens to be the start of a routine.
```

```
;  2285          2410  4                    ! or on data of type ZEM, BPV, or BLV (e.g., if
;  2286          2411  4                    ! the address of a routine is passed as a parameter and
;  2287          2412  4                    ! and a breakpoint is set on that passed-in routine, then
;  2288          2413  4                    ! the Primary describe data of type BPV). These cases
;  2289          2414  4                    ! are not covered by the above "break address" code.
;  2290          2415  4                    !
;  2291          2416  4                    IF NOT .EVENT_ENTRY [EVENT$V_BREAK_ADDRESS]
;  2292          2417  4                    THEN
;  2293          2418  5                        BEGIN
;  2294          2419  5
;  2295          2420  5
;  2296          2421  5                        ! If we can determine from the static address table
;  2297          2422  5                        ! that this is an entry point, then add two to get
;  2298          2423  5                        ! past the entry mask.
;  2299          2424  5                        !
;  2300          2425  5                        IF DBG$IS_IT_ENTRY (.ADDRESS [0])
;  2301          2426  5                        THEN
;  2302          2427  5                            EVENT_ENTRY [EVENT$L_ADDRESS] = .ADDRESS[0] + 2
;  2303          2428  5
;  2304          2429  5                        ELSE
;  2305          2430  6                            BEGIN
;  2306          2431  6
;  2307          2432  6
;  2308          2433  6                            ! If the data type in the value descriptor says
;  2309          2434  6                            ! that we have an entry mask, then add two to
;  2310          2435  6                            ! get past the entry mask.
;  2311          2436  6                            !
;  2312          2437  6                            IF (.VALUE[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_ZEM) OR
;  2313          2438  6                               (.VALUE[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_BPV) OR
;  2314          2439  7                               (.VALUE[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_BLV)
;  2315          2440  6                            THEN
;  2316          2441  6                                EVENT_ENTRY [EVENT$L_ADDRESS] = .ADDRESS[0] + 2;
;  2317          2442  5                            END;
;  2318          2443  4                        END;
;  2319          2444  4
;  2320          2445  4
;  2321          2446  4                    ! Make sure that we can access the value indicated.
;  2322          2447  4                    !
;  2323          2448  4                    DBG$READ_ACCESS (.EVENT_ENTRY [EVENT$L_ADDRESS],
;  2324          2449  5                        (DBG$DATA_LENGTH(EVENT_ENTRY[EVENT$A_VMSDESC]) + 7)
;  2325          2450  4                                                                    ^-3);
;  2326          2451  4
;  2327          2452  4
;  2328          2453  4                    ! Process the event's address additionally based on the
;  2329          2454  4                    ! access type.
;  2330          2455  4                    !
;  2331          2456  4                    SELECTONE .EVENT_ENTRY [EVENT$B_SUB_KIND] OF
;  2332          2457  4                        SET
;  2333          2458  4                        [EVENT$K_ACC_EXEC,
;  2334          2459  4                         EVENT$K_ACC_RTRN]:
;  2335          2460  5                            BEGIN
;  2336          2461  5                            LOCAL
;  2337          2462  5                                EVENT_ADDRESS : REF VECTOR [, BYTE];
;  2338          2463  5
;  2339          2464  5                            EVENT_ADDRESS = .EVENT_ENTRY [EVENT$L_ADDRESS];
;  2340        P 2465  5                            IF NOT WRITE_OPCODE (.EVENT_ADDRESS,
;  2341        P 2466  5                                                 .EVENT_ADDRESS [0]
```

```
                        )
        THEN
            BEGIN
            SIGNAL (DBG$_NOACCESSW, 1, .EVENT_ADDRESS);
            RETURN STS$K_SEVERE;
            END;


        ! Mark the fact We are at /RET .PC, for we want
        ! to take the RET SS$_ROPRAND fault starting
        ! from current PC.  Normally, we won't take
        ! the event if we are at current PC.
        !
        ! IF .EVENT_ADDRESS EQL .DBG$RUNFRAME[DBG$L_USER_PC]
        ! THEN
        !   EVENT_ENTRY [EVENT$V_RET_AT_PC] = TRUE;
        END;

    [EVENT$K_ACC_MDFY]:


        ! Process /MODIFY types by building and saving a non-
        ! volatile value descriptor (i.e. a copy) of the value
        ! indicated.
        !
        BEGIN
        LOCAL
            PRIMARY: REF DBG$PRIMARY;

        ! First, get the value descriptor.
        ! Before we call PRIM_TO_VAL, however, we check
        ! whether the Primary represents an aggregate,
        ! such as an entire array. The scheme of saving
        ! away value descriptors does not work for aggregates,
        ! because we currently do not represent arrays
        ! and records with value descriptors. So, for now
        ! we signal an error saying that we do
        ! not currently support watchpointing of aggregates.
        !
        DBG$NCOPY_DESC(.EVENT_ENTRY[EVENT$L_PRIMARY],
            PRIMARY, DUMMY, FALSE);

        IF .PRIMARY[DBG$V_DHDR_AGGR]
        THEN
            SIGNAL(DBG$_NOWATAGGR);


        ! In order to keep the code path consistency
        ! it is better to turn the volatile value
        ! descriptor into value descriptor.
        !
        DBG$PRIM_TO_VAL (.PRIMARY,
                         DBG$K_V_VALUE_DESC,
                         VALUE);
        DBG$PRIM_TO_VAL (.VALUE,
                         DBG$K_VALUE_DESC,
                         VALUE
```

```
;  2399     2524  5                    );
;  2400     2525  5
;  2401     2526  5                IF .VALUE [DBG$B_DHDR_TYPE] NEQU DBG$K_VALUE_DESC
;  2402     2527  5                THEN
;  2403     2528  5                    SIGNAL(DBG$_WATCHSIZE)
;  2404     2529  5
;  2405     2530  5                ELSE
;  2406     2531  6                    BEGIN
;  2407     2532  6                    LOCAL
;  2408     2533  6                        PAGE_ENTRY :    REF EVENT$PAGE_DESCRIPTOR,
;  2409     2534  6                        PAGE_LIST :     REF VECTOR [, LONG],
;  2410     2535  6                        PAGE_FOUND;
;  2411     2536  6
;  2412     2537  6
;  2413     2538  6                    ! The user is not permitted to set a watchpoint in
;  2414     2539  6                    ! P1 space (or in system space)
;  2415     2540  6                    !
;  2416     2541  6                    IF .EVENT_ENTRY [EVENT$L_ADDRESS] GEQA P1_SPACE
;  2417     2542  6                    THEN
;  2418     2543  7                        BEGIN
;  2419     2544  7                        SIGNAL(DBG$_BADWATCH, 1,
;  2420     2545  7                               .EVENT_ENTRY[EVENT$L_ADDRESS]);
;  2421     2546  7                        RETURN STS$K_SEVERE;
;  2422     2547  6                        END;
;  2423     2548  6
;  2424     2549  6
;  2425     2550  6                    ! The user is not permitted to set a watchpoint on
;  2426     2551  6                    ! a variable bound to either the context registers
;  2427     2552  6                    ! or the user runframe registers.
;  2428     2553  6                    !
;  2429     2554  6                    IF DBG$MAP_TO_REG_ADDR
;  2430     2555  6                            (.EVENT_ENTRY [EVENT$L_ADDRESS], DUMMY)
;  2431     2556  6                    THEN
;  2432     2557  7                        BEGIN
;  2433     2558  7                        SIGNAL (DBG$_NOWATTAR);
;  2434     2559  7                        RETURN STS$K_SEVERE;
;  2435     2560  6                        END;
;  2436     2561  6
;  2437     2562  7                    IF (DBG$RUNFRAME [DBG$L_USER_REGS] LEQA
;  2438     2563  6                        .EVENT_ENTRY [EVENT$L_ADDRESS]) AND
;  2439     2564  7                       (.EVENT_ENTRY [EVENT$L_ADDRESS] LSSA
;  2440     2565  7                        DBG$RUNFRAME [DBG$L_USER_REGS] + 68)
;  2441     2566  6                    THEN
;  2442     2567  7                        BEGIN
;  2443     2568  7                        SIGNAL(DBG$_BADWATCH, 1,
;  2444     2569  7                               .EVENT_ENTRY[EVENT$L_ADDRESS]);
;  2445     2570  7                        RETURN STS$K_SEVERE;
;  2446     2571  6                        END;
;  2447     2572  6
;  2448     2573  6
;  2449     2574  6                    ! Save the value descriptor by making a copy of
;  2450     2575  6                    ! the value descriptor out of non-volatile storage.
;  2451     2576  6                    !
;  2452     2577  6                    ! Obtain the symid list, and lock it.
;  2453     2578  6                    !
;  2454     2579  6                    DBG$NGET_SYMID (.VALUE, SYMID_LIST, DUMMY);
;  2455     2580  6                    DBG$STA_LOCK_SYMID (.SYMID_LIST);
```

```
;  2456          2581  6
;  2457          2582  6
;  2458          2583  6                    ! Copy the value descriptor.
;  2459          2584  6                    !
;  2460          2585  6                    DBG$NCOPY_DESC (.VALUE,
;  2461          2586  6                                    EVENT_ENTRY [EVENT$L_VALDESC],
;  2462          2587  6                                    DUMMY);
;  2463          2588  6
;  2464          2589  6
;  2465          2590  6                    ! Flag that this event has a value descriptor, and
;  2466          2591  6                    ! point the event entry to it's value.
;  2467          2592  6                    !
;  2468          2593  6                    EVENT_ENTRY [EVENT$V_HAS_VAL_DSCR] = TRUE;
;  2469          2594  6
;  2470          2595  6
;  2471          2596  6                    ! Get the pages associated with this value
;  2472          2597  6                    !
;  2473          2598  6                    IF NOT DBG$NGET_PAGES
;  2474          2599  6                              (.EVENT_ENTRY[EVENT$L_PRIMARY],
;  2475          2600  6                               PAGE_LIST, .MESSAGE_VECT)
;  2476          2601  6                    THEN
;  2477          2602  7                        BEGIN
;  2478          2603  7                        SIGNAL(DBG$_BADWATCH, 1,
;  2479          2604  7                                .EVENT_ENTRY[EVENT$L_ADDRESS]);
;  2480          2605  7                        RETURN STS$K_SEVERE;
;  2481          2606  6                        END;
;  2482          2607  6
;  2483          2608  6
;  2484          2609  6                    ! While there are entries in the page list, add a
;  2485          2610  6                    ! new page reference in the page queue.
;  2486          2611  6                    !
;  2487          2612  6                    DO
;  2488          2613  7                        BEGIN
;  2489          2614  7
;  2490          2615  7
;  2491          2616  7                    ! Search through the existing entries in the
;  2492          2617  7                    ! page queue for this page.
;  2493          2618  7                    !
;  2494          2619  7                    PAGE_ENTRY = .EVENT$PAGE_QUEUE [L_QUEUE_FLINK];
;  2495          2620  7                    PAGE_FOUND = FALSE;
;  2496          2621  7                    WHILE .PAGE_ENTRY NEQA EVENT$PAGE_QUEUE DO
;  2497          2622  8                        BEGIN
;  2498          2623  8                        IF .PAGE_ENTRY [EVENT$L_PAGE_ADDRESS] EQLA
;  2499          2624  8                           .PAGE_LIST [1]
;  2500          2625  8                        THEN
;  2501          2626  8
;  2502          2627  8
;  2503          2628  8                            ! A match is found, so bump the refer-
;  2504          2629  8                            ! ence count, flag a find, and exit
;  2505          2630  8                            ! this loop.
;  2506          2631  8                            !
;  2507          2632  9                            BEGIN
;  2508          2633  9                            PAGE_ENTRY [EVENT$W_PAGE_REF_COUNT] =
;  2509          2634  9                                .PAGE_ENTRY [EVENT$W_PAGE_REF_COUNT] + 1;
;  2510          2635  9                            PAGE_FOUND = TRUE;
;  2511          2636  9                            EXITLOOP;
;  2512          2637  9                            END
```

```
;  2513          2638  9
;  2514          2639  9
;  2515          2640  9
;  2516          2641  9
;  2517          2642  8
;  2518          2643  8
;  2519          2644  8
;  2520          2645  7
;  2521          2646  7
;  2522          2647  7
;  2523          2648  7
;  2524          2649  7
;  2525          2650  7
;  2526          2651  7
;  2527          2652  7
;  2528          2653  8
;  2529          2654  8
;  2530          2655  8
;  2531          2656  8
;  2532          2657  8
;  2533          2658  8
;  2534          2659  8
;  2535        P 2660  8                                   ! Not found yet, point to the next one.
;  2536        P 2661  8                                   !
;  2537        P 2662  8                                   ELSE
;  2538          2663  9                                       PAGE_ENTRY = .PAGE_ENTRY [EVENT$L_PAGE_FLINK];
;  2539          2664  8
;  2540          2665  9                                   END;
;  2541          2666  9
;  2542          2667  9                               ! If no match was found, add a new entry to the
;  2543          2668  8                               ! page queue, and fill it in.
;  2544        P 2669  8                               !
;  2545        P 2670  8                               IF NOT .PAGE_FOUND
;  2546          2671  9                               THEN
;  2547          2672  8                                   BEGIN
;  2548          2673  9                                   LOCAL
;  2549          2674  9                                       ADDRESS :   VECTOR [2],
;  2550          2675  9                                       PROTECT;
;  2551          2676  8
;  2552          2677  8                                   ADDRESS [0] = .PAGE_LIST [1];
;  2553          2678  8                                   ADDRESS [1] = .PAGE_LIST [1];
;  2554          2679  8                                   IF NOT $SETPRT (INADR   =   ADDRESS,
;  2555          2680  8                                                   PROT    =   PRT$C_UR,
;  2556          2681  8                                                   PRVPRT  =   PROTECT
;  2557          2682  7                                                  )
;  2558          2683  7                                   THEN
;  2559          2684  7                                       BEGIN
;  2560          2685  6                                       SIGNAL (DBG$_BADWATCH, 1, .PAGE_LIST [1]);
;  2561          2686  5                                       RETURN STS$K_SEVERE;
;  2562          2687  4                                       END;
;  2563          2688  4                                   IF NOT $SETPRT (INADR   =   ADDRESS,
;  2564          2689  4                                                   PROT    =   .PROTECT
;  2565          2690  4                                                  )
;  2566          2691  4                                   THEN
;  2567          2692  4                                       BEGIN
;  2568          2693  4                                       SIGNAL (DBG$_NOWPROT);
;  2569          2694  4                                       RETURN STS$K_SEVERE;
                                                              END;
                                                          PAGE_ENTRY = DBG$GET_MEMORY (EVENT$K_PAGE_DESCRIPTOR_SIZE);
                                                          INSQUE (.PAGE_ENTRY, EVENT$PAGE_QUEUE);
                                                          PAGE_ENTRY [EVENT$L_PAGE_ADDRESS] =
                                                              .PAGE_LIST [1];
                                                          PAGE_ENTRY [EVENT$W_PAGE_REF_COUNT] = 1;
                                                          END;
                                                      PAGE_LIST = .PAGE_LIST [0];
                                                      END
                                                  WHILE .PAGE_LIST;
                                                  END;
                                              END;
                                          TES;

                                      ! Point to the next noun node.
                                      !
                                      NOUN_NODE = .NOUN_NODE [DBG$L_NOUN_LINK];
```

```
;  2570          2695  4                    ! If there's more addresses, get an new event descriptor
;  2571          2696  4                    ! from permanent memory, copy the current one to it, and
;  2572          2697  4                    ! link it to the working queue; otherwise, exit this loop.
;  2573          2698  4                    !
;  2574          2699  4                    IF .NOUN_NODE NEQA 0
;  2575          2700  4                    THEN
;  2576          2701  4                        BEGIN
;  2577          2702  5
;  2578          2703  5                        ! Get another event descriptor, and copy the old
;  2579          2704  5                        ! one into it.
;  2580          2705  5                        !
;  2581          2706  5                        EVENT_ENTRY = DBG$GET_MEMORY
;  2582          2707  5                                      (EVENT$K_EVENT_DESCRIPTOR_SIZE);
;  2583          2708  5                        CH$MOVE
;  2584          2709  5                            (EVENT$K_EVENT_DESCRIPTOR_SIZE * 4,
;  2585          2710  5                             .EVENT_QUEUE [L_QUEUE_FLINK],
;  2586          2711  5                             .EVENT_ENTRY
;  2587          2712  5                            );
;  2588          2713  5
;  2589          2714  5                        ! Link the new one into the working queue.
;  2590          2715  5                        !
;  2591          2716  5                        INSQUE (.EVENT_ENTRY, EVENT_QUEUE);
;  2592          2717  5
;  2593          2718  5                        ! If the old entry has a WHEN, bump the WHEN's
;  2594          2719  5                        ! reference count.
;  2595          2720  5                        !
;  2596          2721  5                        IF .EVENT_ENTRY [EVENT$L_WHEN] NEQA 0
;  2597          2722  5                        THEN
;  2598          2723  5                            WHEN_ENTRY [EVENT$L_WHEN_COUNT] =
;  2599          2724  5                                .WHEN_ENTRY [EVENT$L_WHEN_COUNT] + 1;
;  2600          2725  5
;  2601          2726  5                        ! If the old entry has a DO LIST, bump the DO LIST's
;  2602          2727  5                        ! reference count.
;  2603          2728  5                        !
;  2604          2729  5                        IF .EVENT_ENTRY [EVENT$L_DO] NEQA 0
;  2605          2730  5                        THEN
;  2606          2731  5                            DO_ENTRY [EVENT$L_DO_LIST_COUNT] =
;  2607          2732  5                                .DO_ENTRY [EVENT$L_DO_LIST_COUNT] + 1;
;  2608          2733  5
;  2609          2734  5                        END
;  2610          2735  5
;  2611          2736  4                    ELSE
;  2612          2737  4                        EXITLOOP;
;  2613          2738  4
;  2614          2739  4                    END
;  2615          2740  4
;  2616          2741  2                END;
;  2617          2742  2
;  2618          2743  2            ! Instruction events...
;  2619          2744  2            !
;  2620          2745  2            [EVENT$K_KIND_INS]:
;  2621          2746  2
;  2622          2747  2
;  2623          2748  2
;  2624          2749  2
;  2625          2750  2
;  2626          2751  2
```

```
;  2627          2752  2          SELECTONE .EVENT_ENTRY [EVENT$B_SUB_KIND] OF
;  2628          2753  2              SET
;  2629          2754  2              [EVENT$K_INS_CALL]:
;  2630          2755  2                  EVENT_ENTRY [EVENT$L_OPCODE_LIST] = DBG$OPCODES_CALL;
;  2631          2756  2              [EVENT$K_INS_BRAN]:
;  2632          2757  2                  EVENT_ENTRY [EVENT$L_OPCODE_LIST] = DBG$OPCODES_BRANCH;
;  2633          2758  2              [EVENT$K_INS_EVRY]:
;  2634          2759  2                  EVENT_ENTRY [EVENT$L_OPCODE_LIST] = 0;
;  2635          2760  2              [EVENT$K_INS_USER]:
;  2636          2761  2                  BEGIN
;  2637          2762  2                  LOCAL
;  2638          2763  2                      OPCODE_LIST: REF BITVECTOR[512];
;  2639          2764  2
;  2640          2765  2
;  2641          2766  2                  ! Allocate space for the opcode list.
;  2642          2767  2                  ! 512 bits = 16 longwords
;  2643          2768  2                  !
;  2644          2769  2                  OPCODE_LIST = DBG$GET_MEMORY(16);
;  2645          2770  2
;  2646          2771  2
;  2647          2772  2                  ! If this is a STEP or SET STEP command without any /INST=(opcode...)
;  2648          2773  2                  ! and we get here then the user must have previously
;  2649          2774  2                  ! done SET STEP INST=(opcode-list). In this case we
;  2650          2775  2                  ! initialize the OPCODE_LIST from the copy of
;  2651          2776  2                  ! the opcode list that we saved away on the SET STEP command.
;  2652          2777  2                  ! Otherwise, we just clear the opcode list.
;  2653          2778  2                  !
;  2654          2779  3                  IF (.NOUN_NODE EQLA 0) AND
;  2655          2780  4                     ((.VERB_NODE[DBG$B_VERB_COMPOSITE] EQL EVENT$K_STEP) OR
;  2656          2781  4                      (.VERB_NODE[DBG$B_VERB_COMPOSITE] EQL EVENT$K_SET_STEP))
;  2657          2782  3                  THEN
;  2658          2783  4                      BEGIN
;  2659          2784  4                      CH$MOVE (64, DBG$OPCODES_STEP_USER, .OPCODE_LIST);
;  2660          2785  4                      CH$MOVE (64, DBG$OPCODES_STEP_USER, DBG$OPCODES_USER);
;  2661          2786  4                      END
;  2662          2787  3                  ELSE
;  2663          2788  4                      BEGIN
;  2664          2789  4                      CH$FILL (0, 64, .OPCODE_LIST);
;  2665          2790  4                      CH$FILL (0, 64, DBG$OPCODES_USER);
;  2666          2791  3                      END;
;  2667          2792  3
;  2668          2793  3
;  2669          2794  3                  ! For each opcode (found in the noun node(s) value,
;  2670          2795  3                  ! set the corresponding opcode's bit in the User
;  2671          2796  3                  ! opcode bitmap.
;  2672          2797  3                  !
;  2673          2798  3                  WHILE .NOUN_NODE NEQA 0 DO
;  2674          2799  4                      BEGIN
;  2675          2800  4                      OPCODE_LIST [.NOUN_NODE [DBG$L_NOUN_VALUE]] = 1;
;  2676          2801  4                      DBG$OPCODES_USER [.NOUN_NODE [DBG$L_NOUN_VALUE]] = 1;
;  2677          2802  4                      NOUN_NODE = .NOUN_NODE [DBG$L_NOUN_LINK]
;  2678          2803  3                      END;
;  2679          2804  3
;  2680          2805  3
;  2681          2806  3                  ! Now point the event entry to the user opcode bitmap.
;  2682          2807  3                  !
;  2683          2808  3                  EVENT_ENTRY [EVENT$L_OPCODE_LIST] = .OPCODE_LIST;
```

```
;  2684          2809  2                    END;
;  2685          2810  2
;  2686          2811  2
;  2687          2812  2                [EVENT$K_INS_LINE]:
;  2688          2813  2                    EVENT_ENTRY [EVENT$L_OPCODE_LIST] = 0;
;  2689          2814  2
;  2690          2815  2
;  2691          2816  2                ! An invalid sub-kind, yell !
;  2692          2817  2                !
;  2693          2818  2                [OTHERWISE]:
;  2694          2819  2                    $DBG_ERROR('DBGEVENT\EVENT_SEMANTICS 60');
;  2695          2820  2
;  2696          2821  2                TES;
;  2697          2822  2
;  2698          2823  2
;  2699          2824  2            ! /EXCEPTION kind - do nothing.
;  2700          2825  2            !
;  2701          2826  2            [EVENT$K_KIND_EXC]:
;  2702          2827  2                0;
;  2703          2828  2
;  2704          2829  2            ! An invalid kind, yell !
;  2705          2830  2            !
;  2706          2831  2            [OTHERWISE]:
;  2707          2832  2                $DBG_ERROR('DBGEVENT\EVENT_SEMANTICS 70');
;  2708          2833  2
;  2709          2834  2            TES;
;  2710          2835  2
;  2711          2836  2
;  2712          2837  2        ! If this is a STEP or SET STEP command, process the entry now.
;  2713          2838  2        !
;  2714          2839  2        IF .EVENT_ENTRY [EVENT$B_CMD_TYPE] EQLU EVENT$K_TYPE_STEPS
;  2715          2840  2        THEN
;  2716          2841  3            BEGIN
;  2717          2842  3
;  2718          2843  3
;  2719          2844  3            ! If this is a SET STEP command, just point to the user-defined STEP
;  2720          2845  3            ! definitions.
;  2721          2846  3            !
;  2722          2847  3            IF .VERB_NODE [DBG$B_VERB_COMPOSITE] EQLU EVENT$K_SET_STEP
;  2723          2848  3            THEN
;  2724          2849  3                DBG$SET_STP_LVL (USER_DEF_STEP)
;  2725          2850  3
;  2726          2851  3
;  2727          2852  3            ! This is a STEP command.
;  2728          2853  3            !
;  2729          2854  3            ELSE
;  2730          2855  4                BEGIN
;  2731          2856  4
;  2732          2857  4
;  2733          2858  4                ! This is a STEP command.  Make sure that we're not trying to
;  2734          2859  4                ! step from an exception.
;  2735          2860  4                !
;  2736          2861  4                IF .DBG$GB_EXC_BRE_FLAG
;  2737          2862  4                THEN
;  2738          2863  4                    SIGNAL(DBG$_STEFROEXC)
;  2739          2864  4
;  2740          2865  4
```

```
;  2741          2866  4                ! We are not stepping from an exception, so we can indeed step.'
;  2742          2867  4                !
;  2743          2868  4                ELSE
;  2744          2869  5                    BEGIN
;  2745          2870  5
;  2746          2871  5
;  2747          2872  5                    ! Make sure that we can start up where we are at.
;  2748          2873  5                    !
;  2749          2874  5                    IF NOT PROBER(%REF(0), %REF(1), .DBG$RUNFRAME[DBG$L_USER_PC])
;  2750          2875  5                    THEN
;  2751          2876  5                        SIGNAL(DBG$_BADSTARTPC, 1, .DBG$RUNFRAME[DBG$L_USER_PC])
;  2752          2877  5
;  2753          2878  5
;  2754          2879  5                    ! We can indeed start here, so this STEP is okay.
;  2755          2880  5                    !
;  2756          2881  5                    ELSE
;  2757          2882  6                        BEGIN
;  2758          2883  6
;  2759          2884  6
;  2760          2885  6                        ! Point to STEP override definitions.
;  2761          2886  6                        !
;  2762          2887  6                        DBG$SET_STP_LVL (OVERRIDE_STEP);
;  2763          2888  6
;  2764          2889  6
;  2765          2890  6                        ! Update the step count (probably not needed any more).
;  2766          2891  6                        !
;  2767          2892  6                        DBG$GL_STEP_NUM = .EVENT_ENTRY [EVENT$L_AFTER_COUNT];
;  2768          2893  6
;  2769          2894  6
;  2770          2895  6                    ! Is the user didn't type STEP 0 (a NOP), add this entry
;  2771          2896  6                    ! to the event queue.
;  2772          2897  6                    !
;  2773          2898  6                    IF .EVENT_ENTRY [EVENT$L_AFTER_COUNT] GTRU 0
;  2774          2899  6                    THEN
;  2775          2900  7                        BEGIN
;  2776          2901  7
;  2777          2902  7
;  2778          2903  7                        ! Remove this new entry from the working queue, and
;  2779          2904  7                        ! insert it into HEAD end of the Command Queue (we
;  2780          2905  7                        ! want steps to show up first).
;  2781          2906  7                        !
;  2782          2907  7                        REMQUE (.EVENT_ENTRY, EVENT_ENTRY);
;  2783          2908  7                        INSQUE (.EVENT_ENTRY, EVENT$CMD_QUEUE);
;  2784          2909  7
;  2785          2910  7
;  2786          2911  7                        ! Initialize the queue header within the entry (FLINK1
;  2787          2912  7                        ! and BLINK1) to be empty....
;  2788          2913  7                        !
;  2789          2914  7                        EVENT_ENTRY [EVENT$L_EXC_FLINK] = .EVENT_ENTRY;
;  2790          2915  7                        EVENT_ENTRY [EVENT$L_EXC_BLINK] = .EVENT_ENTRY;
;  2791          2916  7
;  2792          2917  7
;  2793          2918  7                        ! If this is a STEP/RETURN, save the current PC and
;  2794          2919  7                        ! set the saved FP to the current FP (unless we are
;  2795          2920  7                        ! looking at a RET instruction, in which case we're
;  2796          2921  7                        ! going to STEP out one routine level).
;  2797          2922  7                        !
```

```
;  2798          2923  7                 IF .EVENT_ENTRY [EVENT$B_SUB_KIND] EQLU
;  2799          2924  7                    EVENT$K_ACC_RTRN
;  2800          2925  7                 THEN
;  2801          2926  8                     BEGIN
;  2802          2927  8                     BUILTIN PROBER;
;  2803          2928  8                     LOCAL
;  2804          2929  8                         CALL_FRAME      : REF VECTOR[,LONG],
;  2805          2930  8                         STACK_FRAME     : REF BLOCK [,BYTE],
;  2806          2931  8                         INSTRUCTION     : BYTE;
;  2807          2932  8
;  2808          2933  8                     EVENT_ENTRY [EVENT$L_USERS_PC] = .DBG$RUNFRAME [DBG$L_USER_PC];
;  2809          2934  8                     EVENT_ENTRY [EVENT$L_USERS_FP] = .DBG$RUNFRAME [DBG$L_USER_FP];
;  2810          2935  8                     IF PROBER(%REF(0),%REF(1),.DBG$RUNFRAME[DBG$L_USER_PC])
;  2811          2936  8                     THEN
;  2812          2937  9                         BEGIN
;  2813          2938  9                         INSTRUCTION = .(.DBG$RUNFRAME[DBG$L_USER_PC])<0,8,0>;
;  2814          2939 10                         IF (.INSTRUCTION EQLU %X'04')
;  2815          2940  9                         THEN
;  2816          2941 10                             BEGIN
;  2817          2942 10                             STACK_FRAME = .DBG$RUNFRAME[DBG$L_USER_FP];
;  2818          2943 10                             IF PROBER(%REF(0), %REF(20), .STACK_FRAME)
;  2819          2944 10                             THEN
;  2820          2945 10                                 EVENT_ENTRY[EVENT$L_USERS_FP] =
;  2821          2946 10                                     .STACK_FRAME[SF$L_SAVE_FP];
;  2822          2947  9                             END;
;  2823          2948  8                         END;
;  2824          2949  8
;  2825          2950  8                     ! Save more info from FP. in the Event.
;  2826          2951  8                     !
;  2827          2952  8                     CALL_FRAME = DBG$GET_MEMORY(5);
;  2828          2953  8                     EVENT_ENTRY[EVENT$L_CALL_FRAME] = .CALL_FRAME;
;  2829          2954  8                     CH$MOVE(20,.EVENT_ENTRY[EVENT$L_USERS_FP],.CALL_FRAME);
;  2830          2955  7                     END;
;  2831          2956  7
;  2832          2957  7
;  2833          2958  7                     ! This was a STEP command - stop taking commands.
;  2834          2959  7                     !
;  2835          2960  7                     DBG$GB_TAKE_CMD = FALSE;
;  2836          2961  6                     END;
;  2837          2962  6
;  2838          2963  5                 END;
;  2839          2964  5
;  2840          2965  4             END;
;  2841          2966  4
;  2842          2967  3         END;
;  2843          2968  3
;  2844          2969  3
;  2845          2970  3         ! Setup the stepping flags.
;  2846          2971  3         !
;  2847          2972  3         DBG$GB_STP_PTR [EVENT$B_STEPPING_KIND] =
;  2848          2973  3             .EVENT_ENTRY [EVENT$B_SUB_KIND];
;  2849          2974  3         DBG$GB_STP_PTR [EVENT$V_STEPPING_LINE] =
;  2850          2975  3             .EVENT_ENTRY [EVENT$V_STEP_LINE];
;  2851          2976  3         DBG$GB_STP_PTR [EVENT$V_STEPPING_OVER] =
;  2852          2977  3             .EVENT_ENTRY [EVENT$V_STEP_OVER];
;  2853          2978  3         DBG$GB_STP_PTR [EVENT$V_STEPPING_SOURCE] =
;  2854          2979  3             .EVENT_ENTRY [EVENT$V_STEP_SOURCE];
```

```
;  2855          2980  3          DBG$GB_STP_PTR [EVENT$V_STEPPING_NOSYSTEM] =
;  2856          2981  3              .EVENT_ENTRY [EVENT$V_STEP_NOSYSTEM];
;  2857          2982  3          DBG$GB_STP_PTR [EVENT$V_STEPPING_NOSILENT] =
;  2858          2983  3              .EVENT_ENTRY [EVENT$V_STEP_NOSILENT];
;  2859          2984  3
;  2860          2985  3
;  2861          2986  3          ! Save away the opcode list we have built. We get here on
;  2862          2987  3          ! SET STEP INST= (opcode-list)
;  2863          2988  3          !
;  2864          2989  3          IF .VERB_NODE[DBG$B_VERB_COMPOSITE] EQL EVENT$K_SET_STEP
;  2865          2990  3          THEN
;  2866          2991  4              BEGIN
;  2867          2992  4              IF .EVENT_ENTRY[EVENT$B_SUB_KIND] EQL EVENT$K_INS_USER
;  2868          2993  4              THEN
;  2869          2994  5                  BEGIN
;  2870          2995  5                  CH$MOVE(64, DBG$OPCODES_USER, DBG$OPCODES_STEP_USER);
;  2871          2996  5                  DBG$GB_STP_PTR [EVENT$L_STEPPING_OP_LIST] = DBG$OPCODES_STEP_USER;
;  2872          2997  5                  END
;  2873          2998  4
;  2874          2999  4              ELSE
;  2875          3000  4                  DBG$GB_STP_PTR [EVENT$L_STEPPING_OP_LIST] = 0;
;  2876          3001  4              END;
;  2877          3002  3          END
;  2878          3003  3
;  2879          3004  3
;  2880          3005  3      ! This is not a STEP or SET STEP command, so it must be SET BREAK, SET
;  2881          3006  3      ! TRACE, or SET WATCH.
;  2882          3007  3      !
;  2883          3008  3      ELSE
;  2884          3009  3          BEGIN
;  2885          3010  3
;  2886          3011  3
;  2887          3012  3          ! Now that we've built a working list of event entries, add them to
;  2888          3013  3          ! the Command Queue, deleting existing entries that match new ones.
;  2889          3014  3          !
;  2890          3015  3          ! For each new event entry, check the existing entries in the
;  2891          3016  3          ! Command queue for any entries that match (command types, and
;  2892          3017  3          ! addresses if the command type is an access type).
;  2893          3018  3          !
;  2894          3019  3          ! So, while there are (still) entries on the working queue....
;  2895          3020  3          !
;  2896          3021  3          WHILE .EVENT_QUEUE [L_QUEUE_FLINK] NEQA EVENT_QUEUE DO
;  2897          3022  4              BEGIN
;  2898          3023  4
;  2899          3024  4
;  2900          3025  4              ! Point EVENT_ENTRY to the first new event entry on the
;  2901          3026  4              ! working queue.
;  2902          3027  4              !
;  2903          3028  4              EVENT_ENTRY = .EVENT_QUEUE [L_QUEUE_FLINK];
;  2904          3029  4
;  2905          3030  4
;  2906          3031  4              ! Point to the first and next existing entries in the Command Queue.
;  2907          3032  4              !
;  2908          3033  4              QUEUE_ENTRY = .EVENT$CMD_QUEUE [L_QUEUE_FLINK];
;  2909          3034  4              NEXTQ_ENTRY = .QUEUE_ENTRY [EVENT$L_CMD_FLINK];
;  2910          3035  4
;  2911          3036  4
```

```
;  2912          3037  4          ! For each existing entry...
;  2913          3038  4          !
;  2914          3039  4          WHILE .QUEUE_ENTRY NEQA EVENT$CMD_QUEUE DO
;  2915          3040  5              BEGIN
;  2916          3041  5
;  2917          3042  5
;  2918          3043  5              ! Ignore any 'deleted' entries that haven't actually been
;  2919          3044  5              ! removed.
;  2920          3045  5              !
;  2921          3046  5              IF NOT .QUEUE_ENTRY [EVENT$V_DELETED]
;  2922          3047  5              THEN
;  2923          3048  6                  BEGIN
;  2924          3049  6
;  2925          3050  6
;  2926          3051  6                  ! Compare the command kind of this existing entry and the
;  2927          3052  6                  ! new one.
;  2928          3053  6                  !
;  2929          3054  6                  IF .EVENT_ENTRY [EVENT$B_CMD_KIND] EQLU
;  2930          3055  6                     .QUEUE_ENTRY [EVENT$B_CMD_KIND]
;  2931          3056  6                  THEN
;  2932          3057  7                      BEGIN
;  2933          3058  7
;  2934          3059  7
;  2935          3060  7                      ! The command kinds match.  Select on the kind.
;  2936          3061  7                      !
;  2937          3062  7                      SELECTONE .EVENT_ENTRY [EVENT$B_CMD_KIND] OF
;  2938          3063  7                          SET
;  2939          3064  7
;  2940          3065  7
;  2941          3066  7                      ! The command kind is access (/READ, /WRITE,
;  2942          3067  7                      ! /MODIFY, /EXECUTE, or /RETURN).  We now have to
;  2943          3068  7                      ! compare the entries to determine if the existing
;  2944          3069  7                      ! one must be deleted.
;  2945          3070  7                      !
;  2946          3071  7                      [EVENT$K_KIND_ACC]:
;  2947          3072  8                          BEGIN
;  2948          3073  8                          LOCAL CODE_1,CODE_2;
;  2949          3074  8
;  2950          3075  8                          !   If either of the entries is /EXECUTE or
;  2951          3076  8                          ! /RETURN, we'll compare the byte addresses,
;  2952          3077  8                          ! otherwise compare the two VMS descriptors,
;  2953          3078  8                          ! and delete the exisiting entry if they're
;  2954          3079  8                          ! the same.
;  2955          3080  8                          !
;  2956          3081  8                          CODE_1 = .EVENT_ENTRY [EVENT$B_SUB_KIND];
;  2957          3082  8                          CODE_2 = .QUEUE_ENTRY [EVENT$B_SUB_KIND];
;  2958          3083 10                          IF (IF (.CODE_1 EQLU EVENT$K_ACC_EXEC)
;  2959          3084 10                              OR (.CODE_1 EQLU EVENT$K_ACC_RTRN)
;  2960          3085 10                              OR (.CODE_2 EQLU EVENT$K_ACC_EXEC)
;  2961          3086 10                              OR (.CODE_2 EQLU EVENT$K_ACC_RTRN)
;  2962          3087  9                              THEN
;  2963          3088  9                                  .EVENT_ENTRY [EVENT$L_ADDRESS] EQLA
;  2964          3089  9                                  .QUEUE_ENTRY [EVENT$L_ADDRESS]
;  2965          3090  9                              ELSE
;  2966          3091  9                                  COMPARE_VMSDESC(EVENT_ENTRY [EVENT$A_VMSDESC],
;  2967          3092  9                                                  QUEUE_ENTRY [EVENT$A_VMSDESC])
;  2968          3093  9                             )
```

```
;  2969          3094  8                    THEN
;  2970          3095  8                        QUEUE_ENTRY [EVENT$V_DELETED] = TRUE;
;  2971          3096  7                    END;
;  2972          3097  7
;  2973          3098  7                ! The command kind is instruction.  Compare the two
;  2974          3099  7                ! entries' subkind, and delete the existing entry
;  2975          3100  7                ! if they're the same.
;  2976          3101  7                !
;  2977          3102  7                [EVENT$K_KIND_INS]:
;  2978          3103  7                    CASE .EVENT_ENTRY [EVENT$B_SUB_KIND]
;  2979          3104  7                    FROM EVENT$K_INS_CALL TO EVENT$K_INS_USER OF
;  2980          3105  7                        SET
;  2981          3106  7                        [EVENT$K_INS_CALL TO EVENT$K_INS_LINE]:
;  2982          3107  7                            IF .EVENT_ENTRY [EVENT$B_SUB_KIND] EQLU
;  2983          3108  7                               .QUEUE_ENTRY [EVENT$B_SUB_KIND]
;  2984          3109  7                            THEN
;  2985          3110  7                                QUEUE_ENTRY [EVENT$V_DELETED] =
;  2986          3111  7                                    TRUE;
;  2987          3112  7                        [EVENT$K_INS_EVRY TO EVENT$K_INS_USER]:
;  2988          3113  7                            IF (.QUEUE_ENTRY [EVENT$B_SUB_KIND] EQLU
;  2989          3114  8                                EVENT$K_INS_EVRY) OR
;  2990          3115  7                               (.QUEUE_ENTRY [EVENT$B_SUB_KIND] EQLU
;  2991          3116  8                                EVENT$K_INS_USER)
;  2992          3117  8                            THEN
;  2993          3118  7                                QUEUE_ENTRY [EVENT$V_DELETED] =
;  2994          3119  7                                    TRUE;
;  2995          3120  7                        TES;
;  2996          3121  7
;  2997          3122  7                ! The access type is exception, so delete the
;  2998          3123  7                ! existing entry.
;  2999          3124  7                !
;  3000          3125  7                [EVENT$K_KIND_EXC]:
;  3001          3126  7                    QUEUE_ENTRY [EVENT$V_DELETED] = TRUE;
;  3002          3127  7
;  3003          3128  7                ! The command kind is invalid - yell !
;  3004          3129  7                !
;  3005          3130  7                [OTHERWISE]:
;  3006          3131  7                    $DBG_ERROR('DBGEVENT\EVENT_SEMANTICS 80');
;  3007          3132  7
;  3008          3133  7                TES;
;  3009          3134  7
;  3010          3135  7            END;
;  3011          3136  7
;  3012          3137  7        END;
;  3013          3138  6
;  3014          3139  6    ! Point to the next Command Queue entries.
;  3015          3140  5    !
;  3016          3141  5    QUEUE_ENTRY = .NEXTQ_ENTRY;
;  3017          3142  5    NEXTQ_ENTRY = .QUEUE_ENTRY [EVENT$L_CMD_FLINK];
;  3018          3143  5    END;
;  3019          3144  5
;  3020          3145  5! Remove this new entry from the working queue, and insert
;  3021          3146  5
;  3022          3147  4
;  3023          3148  4
;  3024          3149  4
;  3025          3150  4
```

```
;  3026          3151  4              ! it into tail end of the Command Queue.
;  3027          3152  4              !
;  3028          3153  4              REMQUE (.EVENT_ENTRY, EVENT_ENTRY);
;  3029          3154  4              INSQUE (.EVENT_ENTRY, .EVENT$CMD_QUEUE [L_QUEUE_BLINK]);
;  3030          3155  4
;  3031          3156  4
;  3032          3157  4              ! Initialize the queue header within the entry (FLINK1 and BLINK1)
;  3033          3158  4              ! to be empty.
;  3034          3159  4              !
;  3035          3160  4              EVENT_ENTRY [EVENT$L_EXC_FLINK] = .EVENT_ENTRY;
;  3036          3161  4              EVENT_ENTRY [EVENT$L_EXC_BLINK] = .EVENT_ENTRY;
;  3037          3162  3              END;
;  3038          3163  3
;  3039          3164  2          END;
;  3040          3165  2
;  3041          3166  2
;  3042          3167  2      ! Delete any entries that remain in the working queue.
;  3043          3168  2      ! This happens on SET STEP or on STEP 0.
;  3044          3169  2      !
;  3045          3170  2      EVENT_ENTRY = .EVENT_QUEUE[L_QUEUE_FLINK];
;  3046          3171  2      WHILE .EVENT_ENTRY NEQU EVENT_QUEUE DO
;  3047          3172  2          BEGIN
;  3048          3173  3          LOCAL
;  3049          3174  3              NEW_EVENT_ENTRY;
;  3050          3175  3          NEW_EVENT_ENTRY = .EVENT_ENTRY[EVENT$L_CMD_FLINK];
;  3051          3176  3          DELETE_EVENT_ENTRY(.EVENT_ENTRY);
;  3052          3177  3          EVENT_ENTRY = .NEW_EVENT_ENTRY;
;  3053          3178  2          END;
;  3054          3179  2
;  3055          3180  2
;  3056          3181  2      ! All's well that ends well.
;  3057          3182  2      !
;  3058          3183  2      RETURN STS$K_SUCCESS;
;  3059          3184  2
;  3060          3185  1      END;


                                                                .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

54  4E  45  56  45  5C  54  4E  45  56  45  47  42  44  1B  000C1 P.ABL:   .ASCII  <27>\DBGEVENT\<92>\EVENT_SEMANTICS 10\      ;
    30  31  20  53  43  49  54  4E  41  4D  45  53  5F  000D0
54  4E  45  56  45  5C  54  4E  45  56  45  47  42  44  1B  000DD P.ABM:   .ASCII  <27>\DBGEVENT\<92>\EVENT_SEMANTICS 20\      ;
    30  32  20  53  43  49  54  4E  41  4D  45  53  5F  000EC
54  4E  45  56  45  5C  54  4E  45  56  45  47  42  44  1B  000F9 P.ABN:   .ASCII  <27>\DBGEVENT\<92>\EVENT_SEMANTICS 30\      ;
    30  33  20  53  43  49  54  4E  41  4D  45  53  5F  00108
54  4E  45  56  45  5C  54  4E  45  56  45  47  42  44  1B  00115 P.ABO:   .ASCII  <27>\DBGEVENT\<92>\EVENT_SEMANTICS 40\      ;
    30  34  20  53  43  49  54  4E  41  4D  45  53  5F  00124
54  4E  45  56  45  5C  54  4E  45  56  45  47  42  44  1B  00131 P.ABP:   .ASCII  <27>\DBGEVENT\<92>\EVENT_SEMANTICS 50\      ;
    30  35  20  53  43  49  54  4E  41  4D  45  53  5F  00140
54  4E  45  56  45  5C  54  4E  45  56  45  47  42  44  1B  0014D P.ABQ:   .ASCII  <27>\DBGEVENT\<92>\EVENT_SEMANTICS 60\      ;
    30  36  20  53  43  49  54  4E  41  4D  45  53  5F  0015C
54  4E  45  56  45  5C  54  4E  45  56  45  47  42  44  1B  00169 P.ABR:   .ASCII  <27>\DBGEVENT\<92>\EVENT_SEMANTICS 70\      ;
    30  37  20  53  43  49  54  4E  41  4D  45  53  5F  00178
54  4E  45  56  45  5C  54  4E  45  56  45  47  42  44  1B  00185 P.ABS:   .ASCII  <27>\DBGEVENT\<92>\EVENT_SEMANTICS 80\      ;
    30  38  20  53  43  49  54  4E  41  4D  45  53  5F  00194
```

```
                                                        .EXTRN  SYS$SETPRT

                                                        .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                        0FFC 00000      .ENTRY  DBG$EVENT_SEMANTICS, Save R2,R3,R4,R5,R6,-      ; 1840
                                                                R7,R8,R9,R10,R11
                                5E      B4  AE  9E 00002        MOVAB   -76(SP), SP
                             3C AE      3C  AE  9E 00006        MOVAB   EVENT_QUEUE, EVENT_QUEUE                ; 1894
                             40 AE      3C  AE  9E 0000B        MOVAB   EVENT_QUEUE, EVENT_QUEUE+4              ; 1895
                                           10  DD 00010        PUSHL   #16                                     ; 1902
                   00000000G 00            01  FB 00012        CALLS   #1, DBG$GET_MEMORY
                                56         50  D0 00019        MOVL    R0, EVENT_ENTRY
                             3C AE         66  0E 0001C        INSQUE  (EVENT_ENTRY), EVENT_QUEUE              ; 1903
                             08 A6         56  D0 00020        MOVL    EVENT_ENTRY, 8(EVENT_ENTRY)             ; 1904
                             0C A6         56  D0 00024        MOVL    EVENT_ENTRY, 12(EVENT_ENTRY)            ; 1905
                                52      04  AC  D0 00028        MOVL    VERB_NODE, R2                           ; 1910
                                50      01  A2  9A 0002C        MOVZBL  1(R2), R0
                                01         50  91 00030        CMPB    R0, #1                                  ; 1913
                                           05  12 00033        BNEQ    1$
                                14 A6      94  11 00035        CLRB    20(EVENT_ENTRY)                         ; 1914
                                           4C  11 00038        BRB     6$
                                03         50  91 0003A 1$:     CMPB    R0, #3                                  ; 1916
                                           08  12 0003D        BNEQ    2$
                             14 A6   0200  8F  B0 0003F        MOVW    #512, 20(EVENT_ENTRY)                   ; 1918
                                           3F  11 00045        BRB     6$                                      ; 1910
                                0E         50  91 00047 2$:     CMPB    R0, #14                                 ; 1922
                                           06  12 0004A        BNEQ    3$
                             14 A6         01  90 0004C        MOVB    #1, 20(EVENT_ENTRY)                     ; 1923
                                           34  11 00050        BRB     6$
                                11         50  91 00052 3$:     CMPB    R0, #17                                 ; 1925
                                           0A  12 00055        BNEQ    4$
                             14 A6         02  B0 00057        MOVW    #2, 20(EVENT_ENTRY)                     ; 1927
                             16 A6         02  90 0005B        MOVB    #2, 22(EVENT_ENTRY)                     ; 1929
                                           25  11 0005F        BRB     6$
                                0B         50  91 00061 4$:     CMPB    R0, #11                                 ; 1932
                                           0B  1F 00064        BLSSU   5$
                                0C         50  91 00066        CMPB    R0, #12
                                           06  1A 00069        BGTRU   5$
                             14 A6         03  90 0006B        MOVB    #3, 20(EVENT_ENTRY)                     ; 1934
                                           15  11 0006F        BRB     6$
                                   00000000' EF  9F 00071 5$:     PUSHAB  P.ABL                                   ; 1937
                                           01  DD 00077        PUSHL   #1
                                00028362   8F  DD 00079        PUSHL   #164706
                         00000000G 00              03  FB 0007F        CALLS   #3, LIB$SIGNAL
                                      17   A6  94 00086 6$:     CLRB    23(EVENT_ENTRY)                         ; 1945
                                50    18   A6  9E 00089        MOVAB   24(EVENT_ENTRY), R0                     ; 1951
                                                60  D4 0008D        CLRL    (R0)
                                03    14   A6  91 0008F        CMPB    20(EVENT_ENTRY), #3                     ; 1952
                                                3A  12 00093        BNEQ    7$
                                51 00000000G    00  D0 00095        MOVL    DBG$GB_STP_PTR, R1                      ; 1955
                      60 01     01       01   A1  F0 0009C        INSV    1(R1), #1, #1, (R0)
                      53 61     01            09  EF 000A2        EXTZV   #9, #1, (R1), R3                        ; 1956
                      60 01     03            53  F0 000A7        INSV    R3, #3, #1, (R0)
                      53 61     01            0A  EF 000AC        EXTZV   #10, #1, (R1), R3                       ; 1958
                      60 01     05            53  F0 000B1        INSV    R3, #5, #1, (R0)
                                60            10  8A 000B6        BICB2   #16, (R0)                               ; 1959
                      53 61     01            0B  EF 000B9        EXTZV   #11, #1, (R1), R3                       ; 1961
```

```
                60     01           07          53  F0 000BE          INSV    R3, #7, #1, (R0)                        ;
                53     61           01          0C  EF 000C3          EXTZV   #12, #1, (R1), R3                       ; 1963
                60     01           09          53  F0 000C8          INSV    R3, #9, #1, (R0)                        ;
                                                05  11 000CD          BRB     8$                                      ; 1952
                                    60  02BA     8F  AA 000CF 7$:      BICW2   #698, (R0)                              ; 1977
                               1C   A6          01  7D 000D4 8$:      MOVQ    #1, 28(EVENT_ENTRY)                     ; 1988
                                    A6      24  7C  7C 000D8          CLRQ    36(EVENT_ENTRY)                         ; 1994
                                    A2      5A  08  A2 000DB          MOVL    8(R2), NOUN_NODE                        ; 1999
                                    A2      52  04  A2 000DF          MOVL    4(R2), ADVERB_NODE                      ; 2004
                                                03  12 000E3 9$:      BNEQ    10$                                     ; 2009
                                                0213 31 000E5          BRW     51$
                                                62  95 000E8 10$:     TSTB    (ADVERB_NODE)                           ; 2022
                                                07  12 000EA          BNEQ    11$
                               1C   A6      04  A2  D0 000EC          MOVL    4(ADVERB_NODE), 28(EVENT_ENTRY)         ; 2024
                                                5D  11 000F1          BRB     16$                                     ; 2023
                                    01          62  91 000F3 11$:     CMPB    (ADVERB_NODE), #1                       ; 2029
                                                06  12 000F6          BNEQ    12$
                               17   A6          04  88 000F8          BISB2   #4, 23(EVENT_ENTRY)                     ; 2030
                                                52  11 000FC          BRB     16$
                                    02          62  91 000FE 12$:     CMPB    (ADVERB_NODE), #2                       ; 2035
                                                0A  12 00101          BNEQ    13$
                               17   A6          01  88 00103          BISB2   #1, 23(EVENT_ENTRY)                     ; 2037
                               19   A6          02  8A 00107          BICB2   #2, 25(EVENT_ENTRY)                     ; 2038
                                                0D  11 0010B          BRB     14$                                     ; 2039
                                    03          62  91 0010D 13$:     CMPB    (ADVERB_NODE), #3                       ; 2045
                                                0E  12 00110          BNEQ    15$
                               17   A6          01  8A 00112          BICB2   #1, 23(EVENT_ENTRY)                     ; 2047
                               19   A6          02  88 00116          BISB2   #2, 25(EVENT_ENTRY)                     ; 2048
                               19   A6          01  88 0011A 14$:     BISB2   #1, 25(EVENT_ENTRY)                     ; 2049
                                                62  11 0011E          BRB     18$                                     ; 2016
                                    04          62  91 00120 15$:     CMPB    (ADVERB_NODE), #4                       ; 2055
                                                2D  12 00123          BNEQ    17$
                                                04  DD 00125          PUSHL   #4                                      ; 2063
                00000000G      00               01  FB 00127          CALLS   #1, DBG$GET_MEMORY
                                    58          50  D0 0012E          MOVL    R0, WHEN_ENTRY
                00000000'      EF               68  0E 00131          INSQUE  (WHEN_ENTRY), EVENT$WHEN_QUEUE          ; 2064
                                20   A6          58  D0 00138          MOVL    WHEN_ENTRY, 32(EVENT_ENTRY)             ; 2069
                                    53      04  A2  D0 0013C          MOVL    4(ADVERB_NODE), STRING                  ; 2074
                                    50          63  3C 00140          MOVZWL  (STRING), LENGTH                        ; 2075
                                01  A043       0D  90 00143          MOVB    #13, 1(LENGTH)[STRING]                  ; 2076
                                08   A8          01  D0 00148          MOVL    #1, 8(WHEN_ENTRY)                       ; 2081
                                0C   A8          53  D0 0014C          MOVL    STRING, 12(WHEN_ENTRY)                  ; 2082
                                                30  11 00150 16$:     BRB     18$                                     ; 2016
                                    05          62  91 00152 17$:     CMPB    (ADVERB_NODE), #5                       ; 2088
                                                2D  12 00155          BNEQ    19$
                                                04  DD 00157          PUSHL   #4                                      ; 2097
                00000000G      00               01  FB 00159          CALLS   #1, DBG$GET_MEMORY
                                    59          50  D0 00160          MOVL    R0, DO_ENTRY
                00000000'      EF               69  0E 00163          INSQUE  (DO_ENTRY), EVENT$DO_LIST_QUEUE         ; 2098
                                24   A6          59  D0 0016A          MOVL    DO_ENTRY, 36(EVENT_ENTRY)               ; 2103
                                    53      04  A2  D0 0016E          MOVL    4(ADVERB_NODE), STRING                  ; 2108
                                    50          63  3C 00172          MOVZWL  (STRING), LENGTH                        ; 2109
                                01  A043       0D  90 00175          MOVB    #13, 1(LENGTH)[STRING]                  ; 2110
                                08   A9          01  D0 0017A          MOVL    #1, 8(DO_ENTRY)                         ; 2115
                                0C   A9          53  D0 0017E          MOVL    STRING, 12(DO_ENTRY)                    ; 2116
                                                64  11 00182 18$:     BRB     27$                                     ; 2016
                                    06          62  91 00184 19$:     CMPB    (ADVERB_NODE), #6                       ; 2123
```

```
                                    54  1F 00187          BLSSU   26$                                                    ;
                              0A    62  91 00189          CMPB    (ADVERB_NODE), #10                                     ;
                                    4F  1A 0018C          BGTRU   26$                                                    ;
                           15   A6    94 0018E          CLRB    21(EVENT_ENTRY)                                        ; 2130
                              06    62  91 00191          CMPB    (ADVERB_NODE), #6                                      ; 2138
                                    04  12 00194          BNEQ    20$                                                    ;
                                    50  D4 00196          CLRL    R0                                                     ;
                                    3D  11 00198          BRB     25$                                                    ;
                              07    62  91 0019A 20$:     CMPB    (ADVERB_NODE), #7                                      ; 2140
                                    05  12 0019D          BNEQ    21$                                                    ;
                              50    01  D0 0019F          MOVL    #1, R0                                                 ;
                                    33  11 001A2          BRB     25$                                                    ;
                              08    62  91 001A4 21$:     CMPB    (ADVERB_NODE), #8                                      ; 2142
                                    05  12 001A7          BNEQ    22$                                                    ;
                              50    02  D0 001A9          MOVL    #2, R0                                                 ;
                                    29  11 001AC          BRB     25$                                                    ;
                              09    62  91 001AE 22$:     CMPB    (ADVERB_NODE), #9                                      ; 2144
                                    05  12 001B1          BNEQ    23$                                                    ;
                              50    03  D0 001B3          MOVL    #3, R0                                                 ;
                                    1F  11 001B6          BRB     25$                                                    ;
                              0A    62  91 001B8 23$:     CMPB    (ADVERB_NODE), #10                                     ; 2146
                                    05  12 001BB          BNEQ    24$                                                    ;
                              50    04  D0 001BD          MOVL    #4, R0                                                 ;
                                    15  11 001C0          BRB     25$                                                    ;
                     00000000'      EF  9F 001C2 24$:     PUSHAB  P.ABM                                                  ; 2149
                                    01  DD 001C8          PUSHL   #1                                                     ;
                     00028362       8F  DD 001CA          PUSHL   #164706                                                ;
                00000000G   00      03  FB 001D0          CALLS   #3, LIB$SIGNAL                                         ;
                     16   A6        50  90 001D7 25$:     MOVB    R0, 22(EVENT_ENTRY)                                    ; 2136
                                    0B  11 001DB          BRB     27$                                                    ; 2016
                              0B    62  91 001DD 26$:     CMPB    (ADVERB_NODE), #11                                     ; 2157
                                    09  12 001E0          BNEQ    28$                                                    ;
                        15   A6 0F02  8F  B0 001E2          MOVW    #3842, 21(EVENT_ENTRY)                                 ; 2159
                                0109  31  001E8 27$:     BRW     50$                                                    ; 2016
                              0A    62  91 001EB 28$:     CMPB    (ADVERB_NODE), #10                                     ; 2167
                                    0D  13 001EE          BEQL    31$                                                    ;
                              12    62  91 001F0          CMPB    (ADVERB_NODE), #18                                     ;
                                    03  1E 001F3          BGEQU   30$                                                    ;
                                0096  31  001F5 29$:     BRW     40$                                                    ;
                              16    62  91 001F8 30$:     CMPB    (ADVERB_NODE), #22                                     ;
                                    F8  1A 001FB          BGTRU   29$                                                    ;
                        15   A6    01  90 001FD 31$:     MOVB    #1, 21(EVENT_ENTRY)                                    ; 2182
                                    12  91 00201          CMPB    (ADVERB_NODE), #18                                     ; 2190
                                    05  12 00204          BNEQ    32$                                                    ;
                              50    05  D0 00206          MOVL    #5, R0                                                 ;
                                    3D  11 00209          BRB     37$                                                    ;
                              13    62  91 0020B 32$:     CMPB    (ADVERB_NODE), #19                                     ; 2192
                                    05  12 0020E          BNEQ    33$                                                    ;
                              50    06  D0 00210          MOVL    #6, R0                                                 ;
                                    33  11 00213          BRB     37$                                                    ;
                              14    62  91 00215 33$:     CMPB    (ADVERB_NODE), #20                                     ; 2194
                                    05  12 00218          BNEQ    34$                                                    ;
                              50    07  D0 0021A          MOVL    #7, R0                                                 ;
                                    29  11 0021D          BRB     37$                                                    ;
                              16    62  91 0021F 34$:     CMPB    (ADVERB_NODE), #22                                     ; 2196
                                    05  12 00222          BNEQ    35$                                                    ;
                              50    09  D0 00224          MOVL    #9, R0                                                 ;
```

```
                                          1F  11 00227       BRB     37$                             ;
                   15                     62  91 00229 35$:  CMPB    (ADVERB_NODE), #21              ; 2198
                                          05  12 0022C       BNEQ    36$                             ;
                   50                     08  D0 0022E       MOVL    #8, R0                          ;
                                          15  11 00231       BRB     37$                             ;
                           00000000'      EF  9F 00233 36$:  PUSHAB  P.ABN                           ; 2201
                                          01  DD 00239       PUSHL   #1                              ;
                           00028362       8F  DD 0023B       PUSHL   #164706                         ;
              00000000G    00             03  FB 00241       CALLS   #3, LIB$SIGNAL                  ;
                     16    A6             50  90 00248 37$:  MOVB    R0, 22(EVENT_ENTRY)             ; 2188
                                          50  D4 0024C       CLRL    R0                              ; 2209
                   07             16      A6  91 0024E       CMPB    22(EVENT_ENTRY), #7             ;
                                          02  12 00252       BNEQ    38$                             ;
                                          50  D6 00254       INCL    R0                              ;
  18   A6           01     01             50  F0 00256 38$:  INSV    R0, #1, #1, 24(EVENT_ENTRY)     ;
                     19    A6             04  8A 0025C       BICB2   #4, 25(EVENT_ENTRY)             ; 2214
                                          50  D4 00260       CLRL    STATUS                          ; 2225
                                          6E  D4 00262       CLRL    MODRST                          ; 2226
                   07             16      A6  91 00264       CMPB    22(EVENT_ENTRY), #7             ; 2227
                                          1B  12 00268       BNEQ    39$                             ;
                                          5E  DD 0026A       PUSHL   SP                              ; 2234
                                  38      A6  9F 0026C       PUSHAB  56(EVENT_ENTRY)                 ;
                                  34      A6  9F 0026F       PUSHAB  52(EVENT_ENTRY)                 ; 2233
                                  10      AE  9F 00272       PUSHAB  STMTNO                          ; 2229
                                  18      AE  9F 00275       PUSHAB  LINENO                          ;
                           00000000G      00  DD 00278       PUSHL   DBG$RUNFRAME+64                 ; 2234
              00000000G    00             06  FB 0027E       CALLS   #6, DBG$PC_TO_LINE_LOOKUP       ;
                           6C             50  E8 00285 39$:  BLBS    STATUS, 50$                     ; 2236
                     34    A6             01  7D 00288       MOVQ    #1, 52(EVENT_ENTRY)             ; 2239
                                          66  11 0028C       BRB     50$                             ; 2016
                           0C             62  91 0028E 40$:  CMPB    (ADVERB_NODE), #12              ; 2247
                                          06  12 00291       BNEQ    41$                             ;
                     18    A6             20  88 00293       BISB2   #32, 24(EVENT_ENTRY)            ; 2249
                                          09  11 00297       BRB     42$                             ; 2250
                           0D             62  91 00299 41$:  CMPB    (ADVERB_NODE), #13              ; 2256
                                          0A  12 0029C       BNEQ    43$                             ;
                     18    A6             20  8A 0029E       BICB2   #32, 24(EVENT_ENTRY)            ; 2258
                     18    A6             10  88 002A2 42$:  BISB2   #16, 24(EVENT_ENTRY)            ; 2259
                                          4C  11 002A6       BRB     50$                             ; 2016
                           0E             62  91 002A8 43$:  CMPB    (ADVERB_NODE), #14              ; 2265
                                          07  12 002AB       BNEQ    44$                             ;
                     18    A6     80      8F  8A 002AD       BICB2   #128, 24(EVENT_ENTRY)           ; 2267
                                          0A  11 002B2       BRB     45$                             ; 2268
                           0F             62  91 002B4 44$:  CMPB    (ADVERB_NODE), #15              ; 2274
                                          0C  12 002B7       BNEQ    46$                             ;
                     18    A6     80      8F  88 002B9       BISB2   #128, 24(EVENT_ENTRY)           ; 2276
                     18    A6     40      8F  88 002BE 45$:  BISB2   #64, 24(EVENT_ENTRY)            ; 2277
                                          2F  11 002C3       BRB     50$                             ; 2016
                           10             62  91 002C5 46$:  CMPB    (ADVERB_NODE), #16              ; 2283
                                          06  12 002C8       BNEQ    47$                             ;
                     18    A6             08  8A 002CA       BICB2   #8, 24(EVENT_ENTRY)             ; 2285
                                          09  11 002CE       BRB     48$                             ; 2286
                           11             62  91 002D0 47$:  CMPB    (ADVERB_NODE), #17              ; 2292
                                          0A  12 002D3       BNEQ    49$                             ;
                     18    A6             08  88 002D5       BISB2   #8, 24(EVENT_ENTRY)             ; 2294
                     18    A6             04  88 002D9 48$:  BISB2   #4, 24(EVENT_ENTRY)             ; 2295
                                          15  11 002DD       BRB     50$                             ; 2016
```

```
                        00000000'  EF  9F 002DF 49$:    PUSHAB  P.ABO                                   ;2302
                                   01  DD 002E5         PUSHL   #1
                        00028362   8F  DD 002E7         PUSHL   #164706
00000000G   00                     03  FB 002ED         CALLS   #3, LIB$SIGNAL
                52          08     A2  D0 002F4 50$:    MOVL    8(ADVERB_NODE), ADVERB_NODE             ;2309
                                FDE8  31 002F8         BRW     9$                                      ;2009
                50          15     A6  9A 002FB 51$:    MOVZBL  21(EVENT_ENTRY), R0                     ;2316
                                   03  13 002FF         BEQL    52$                                     ;2322
                                 0307  31 00301         BRW     84$
                03          14     A6  91 00304 52$:    CMPB    20(EVENT_ENTRY), #3                     ;2324
                                   03  12 00308         BNEQ    53$
                                 03C8  31 0030A         BRW     101$
                                   5A  D5 0030D 53$:    TSTL    NOUN_NODE                               ;2334
                                   15  12 0030F         BNEQ    54$
                        00000000'  EF  9F 00311         PUSHAB  P.ABP                                   ;2336
                                   01  DD 00317         PUSHL   #1
                        00028362   8F  DD 00319         PUSHL   #164706
00000000G   00                     03  FB 0031F         CALLS   #3, LIB$SIGNAL
                5B                 6A  D0 00326 54$:    MOVL    (NOUN_NODE), PRIMARY                    ;2348
                            20     AE  9F 00329         PUSHAB  DUMMY                                   ;2353
                            20     AE  9F 0032C         PUSHAB  SYMID_LIST
                            5B     DD  0032F            PUSHL   PRIMARY
00000000G   00                     03  FB 00331         CALLS   #3, DBG$NGET_SYMID
                            1C     AE  DD 00338         PUSHL   SYMID_LIST                              ;2354
00000000G   00                     01  FB 0033B         CALLS   #1, DBG$STA_LOCK_SYMID
                                   01  DD 00342         PUSHL   #1                                      ;2360
                            24     AE  9F 00344         PUSHAB  DUMMY
                            28     A6  9F 00347         PUSHAB  40(EVENT_ENTRY)
                            5B     DD  0034A            PUSHL   PRIMARY
00000000G   00                     04  FB 0034C         CALLS   #4, DBG$NCOPY_DESC
                00000000G   00         04  FB 0034C    CALLS   #4, DBG$NCOPY_DESC
                                   18  AE  9F 00353     PUSHAB  VALUE                                   ;2368
                               7E  83  8F  9A 00356     MOVZBL  #131, -(SP)
                            5B     DD  0035A            PUSHL   PRIMARY
                00000000G   00         03  FB 0035C    CALLS   #3, DBG$PRIM_TO_VAL
                               57      18  AE  D0 00363 MOVL    VALUE, R7                               ;2369
        34  A6          14     A7                      MOVC3   #12, 20(R7), 52(EVENT_ENTRY)            ;2370
                            19     A6      20  8A 0036D BICB2   #32, 25(EVENT_ENTRY)                   ;2381
                                       14  A6  95 00371 TSTB    20(EVENT_ENTRY)                         ;2382
                                       06  13 00374     BEQL    55$
                                   01      14  A6  91 00376 CMPB    20(EVENT_ENTRY), #1                 ;2383
                                       1D  12 0037A     BNEQ    57$
                                   04      16  A6  91 0037C 55$:    CMPB    22(EVENT_ENTRY), #4         ;2385
                                       17  13 00380     BEQL    57$
                            79     8F      02  AB  91 00382 CMPB    2(PRIMARY), #121                    ;2388
                                       10  12 00387     BNEQ    57$
                                   02      07  AB  91 00389 CMPB    7(PRIMARY), #2                      ;2390
                                       06  13 0038D     BEQL    56$
                                   08      07  AB  91 0038F CMPB    7(PRIMARY), #8                      ;2391
                                       04  12 00393     BNEQ    57$
                            19     A6          20  88 00395 56$:    BISB2   #32, 25(EVENT_ENTRY)        ;2393
                                       08  AC  DD 00399 57$:    PUSHL   MESSAGE_VECT                    ;2400
7E      19  A6                     01      05  EF 0039C     EXTZV   #5, #1, 25(EVENT_ENTRY), -(SP)
                                       14  AE  9F 003A2     PUSHAB  ADR_TYP                             ;2399
                                       40  AE  9F 003A5     PUSHAB  ADDRESS
                                       5B  DD  003A8        PUSHL   PRIMARY
                00000000G   00         05  FB 003AA    CALLS   #5, DBG$NGET_ADDRESS
                                   03      50  E8 003B1     BLBS    R0, 58$
```

```
                                     01F1   31 003B4         BRW      78$                                   ;
                   52          2C      A6   9E 003B7 58$:    MOVAB    44(EVENT_ENTRY), R2                   ; 2403
                   62          34      AE   D0 003BB         MOVL     ADDRESS, (R2)                         ;
        24     19  A6                  05   E0 003BF         BBS      #5, 25(EVENT_ENTRY), 60$              ; 2416
                               34      AE   DD 003C4         PUSHL    ADDRESS                               ; 2425
          00000000G  00                01   FB 003C7         CALLS    #1, DBG$IS_IT_ENTRY                   ;
                   12                  50   E8 003CE         BLBS     R0, 59$                               ;
                   17          16      A7   91 003D1         CMPB     22(R7), #23                           ; 2437
                                       0C   13 003D5         BEQL     59$                                   ;
                   20          16      A7   91 003D7         CMPB     22(R7), #32                           ; 2438
                                       06   13 003DB         BEQL     59$                                   ;
                   21          16      A7   91 003DD         CMPB     22(R7), #33                           ; 2439
                                       05   12 003E1         BNEQ     60$                                   ;
        62     34  AE                  02   C1 003E3 59$:    ADDL3    #2, ADDRESS, (R2)                     ; 2441
                               34      A6   9F 003E8 60$:    PUSHAB   52(EVENT_ENTRY)                       ; 2449
          00000000G  00                01   FB 003EB         CALLS    #1, DBG$DATA_LENGTH                   ;
                   50                  07   C0 003F2         ADDL2    #7, R0                                ;
        7E         50          FD      8F   78 003F5         ASHL     #-3, R0, -(SP)                        ; 2450
                                       62   DD 003FA         PUSHL    (R2)                                  ; 2448
          00000000G  00                02   FB 003FC         CALLS    #2, DBG$READ_ACCESS                   ;
                   50          16      A6   9A 00403         MOVZBL   22(EVENT_ENTRY), R0                   ; 2456
                   03                  50   91 00407         CMPB     R0, #3                                ; 2458
                                       29   1F 0040A         BLSSU    61$                                   ;
                   04                  50   91 0040C         CMPB     R0, #4                                ;
                                       24   1A 0040F         BGTRU    61$                                   ;
                   53                  62   D0 00411         MOVL     (R2), EVENT_ADDRESS                   ; 2464
               10  AE                  63   9A 00414         MOVZBL   (EVENT_ADDRESS), OP                   ; 2467
                                       01   DD 00418         PUSHL    #1                                    ;
                               14      AE   9F 0041A         PUSHAB   OP                                    ;
                                       53   DD 0041D         PUSHL    EVENT_ADDRESS                         ;
          00000000G  00                03   FB 0041F         CALLS    #3, DBG$WRITE_MEM                     ;
                   72                  50   E8 00426         BLBS     R0, 63$                               ;
                                       53   DD 00429         PUSHL    EVENT_ADDRESS                         ; 2470
                                       01   DD 0042B         PUSHL    #1                                    ;
                        00028220       8F   DD 0042D         PUSHL    #164384                               ;
                                       7C   11 00433         BRB      67$                                   ;
                   02                  50   91 00435 61$:    CMPB     R0, #2                                ; 2485
                                       61   12 00438         BNEQ     63$                                   ;
                                       7E   D4 0043A         CLRL     -(SP)                                 ; 2506
                               24      AE   9F 0043C         PUSHAB   DUMMY                                 ;
                               1C      AE   9F 0043F         PUSHAB   PRIMARY                               ;
                               28      A6   DD 00442         PUSHL    40(EVENT_ENTRY)                       ;
          00000000G  00                04   FB 00445         CALLS    #4, DBG$NCOPY_DESC                    ;
                   53          14      AE   D0 0044C         MOVL     PRIMARY, R3                           ; 2509
                   0D          04      A3   E9 00450         BLBC     4(R3), 62$                            ;
                        00028A4A       8F   DD 00454         PUSHL    #166474                               ; 2511
          00000000G  00                01   FB 0045A         CALLS    #1, LIB$SIGNAL                        ;
                               18      AE   9F 00461 62$:    PUSHAB   VALUE                                 ; 2518
                   7E          83      8F   9A 00464         MOVZBL   #131, -(SP)                           ;
                                       53   DD 00468         PUSHL    R3                                    ;
          00000000G  00                03   FB 0046A         CALLS    #3, DBG$PRIM_TO_VAL                   ;
                               18      AE   9F 00471         PUSHAB   VALUE                                 ; 2521
                   7E          7A      8F   9A 00474         MOVZBL   #122, -(SP)                           ;
                               20      AE   DD 00478         PUSHL    VALUE                                 ;
          00000000G  00                03   FB 0047B         CALLS    #3, DBG$PRIM_TO_VAL                   ;
0000007A   8F     18  BE           08  10   ED 00482         CMPZV    #16, #8, @VALUE, #122                 ; 2526
                                       10   13 0048C         BEQL     64$                                   ;
```

```
                00028A52   8F   DD 0048E             PUSHL   #166482                                    ;2528
00000000G   00             01   FB 00494             CALLS   #1, LIB$SIGNAL
                      013D 31   0049B 63$:            BRW     81$
40000000    8F             62   D1 0049E 64$:        CMPL    (R2), #1073741824                         ;2541
                           14   1F 004A5             BLSSU   68$
                           62   DD 004A7 65$:        PUSHL   (R2)                                      ;2545
                           01   DD 004A9 66$:        PUSHL   #1                                        ;2544
                00028230   8F   DD 004AB             PUSHL   #164400
00000000G   00             03   FB 004B1 67$:        CALLS   #3, LIB$SIGNAL
                      00ED 31   004B8                 BRW     78$                                       ;2546
                        20 AE   9F 004BB 68$:        PUSHAB  DUMMY                                     ;2555
                           62   DD 004BE             PUSHL   (R2)
00000000G   00             02   FB 004C0             CALLS   #2, DBG$MAP_TO_REG_ADDR
                09         50   E9 004C7             BLBC    R0, 69$
                00028CA8   8F   DD 004CA             PUSHL   #167080                                   ;2558
                      00CE 31   004D0                 BRW     77$
                50 00000000G 00  9E 004D3 69$:        MOVAB   DBG$RUNFRAME+4, R0                        ;2562
                50         62   D1 004DA             CMPL    (R2), R0
                           0C   1F 004DD             BLSSU   70$
                50 00000000G 00  9E 004DF             MOVAB   DBG$RUNFRAME+72, R0                       ;2565
                50         62   D1 004E6             CMPL    (R2), R0
                           BC   1F 004E9             BLSSU   65$
                        20 AE   9F 004EB 70$:        PUSHAB  DUMMY                                     ;2579
                        20 AE   9F 004EE             PUSHAB  SYMID_LIST
                        20 AE   DD 004F1             PUSHL   VALUE
00000000G   00             03   FB 004F4             CALLS   #3, DBG$NGET_SYMID
                        1C AE   DD 004FB             PUSHL   SYMID_LIST                                ;2580
00000000G   00             01   FB 004FE             CALLS   #1, DBG$STA_LOCK_SYMID
                        20 AE   9F 00505             PUSHAB  DUMMY                                     ;2586
                        30 A6   9F 00508             PUSHAB  48(EVENT_ENTRY)
                        20 AE   DD 0050B             PUSHL   VALUE
00000000G   00             03   FB 0050E             CALLS   #3, DBG$NCOPY_DESC
              17 A6            08   88 00515        BISB2   #8, 23(EVENT_ENTRY)                         ;2593
                        08 AC   DD 00519             PUSHL   MESSAGE_VECT                              ;2600
                        28 AE   9F 0051C             PUSHAB  PAGE_LIST                                 ;2599
                        28 A6   DD 0051F             PUSHL   40(EVENT_ENTRY)
00000000G   00             03   FB 00522             CALLS   #3, DBG$NGET_PAGES
                03         50   E8 00529             BLBS    R0, 71$
                      FF78 31   0052C                 BRW     65$
                50       24 AE   D0 0052F 71$:        MOVL    PAGE_LIST, R4                             ;2624
                52 00000000' EF   D0 00533 72$:        MOVL    EVENT$PAGE_QUEUE, PAGE_ENTRY              ;2619
                           55   D4 0053A             CLRL    PAGE_FOUND                                ;2620
                50 00000000' EF   9E 0053C 73$:        MOVAB   EVENT$PAGE_QUEUE, R0                      ;2621
                50         52   D1 00543             CMPL    PAGE_ENTRY, R0
                           13   13 00546             BEQL    75$
              04 A4      08 A2   D1 00548             CMPL    8(PAGE_ENTRY), 4(R4)                      ;2624
                           08   12 0054D             BNEQ    74$
                        0C A2   B6 0054F             INCW    12(PAGE_ENTRY)                            ;2634
                55         01   D0 00552             MOVL    #1, PAGE_FOUND                            ;2635
                           05   11 00555             BRB     75$                                       ;2632
                52         62   D0 00557 74$:        MOVL    (PAGE_ENTRY), PAGE_ENTRY                  ;2643
                           E0   11 0055A             BRB     73$                                       ;2621
                6D         55   E8 0055C 75$:        BLBS    PAGE_FOUND, 80$                           ;2651
                53       24 AE   D0 0055F             MOVL    PAGE_LIST, R3                             ;2658
              2C AE      04 A3   D0 00563             MOVL    4(R3), ADDRESS
              30 AE      04 A3   D0 00568             MOVL    4(R3), ADDRESS+4                          ;2659
                        28 AE   9F 0056D             PUSHAB  PROTECT                                   ;2663
```

```
                                    0F   DD 00570          PUSHL   #15                                                    ;
                                    7E   7C 00572          CLRQ    -(SP)                                                  ;
                                 3C AE   9F 00574          PUSHAB  ADDRESS                                                ;
                00000000G        00      FB 00577          CALLS   #5, SYS$SETPRT                                         ;
                                 06      E8 0057E          BLBS    R0, 76$                                                ;
                                 04 A3   DD 00581          PUSHL   4(R3)                                                  ; 2666
                                    FF22 31 00584          BRW     66$                                                    ;
                                    7E   D4 00587 76$:     CLRL    -(SP)                                                  ; 2671
                                 2C AE   DD 00589          PUSHL   PROTECT                                                ;
                                    7E   7C 0058C          CLRQ    -(SP)                                                  ;
                                 3C AE   9F 0058E          PUSHAB  ADDRESS                                                ;
                00000000G        00      FB 00591          CALLS   #5, SYS$SETPRT                                         ;
                                 11      E8 00598          BLBS    R0, 79$                                                ;
                           000284C4 8F   DD 0059B          PUSHL   #165060                                                ; 2674
                00000000G        00      FB 005A1 77$:     CALLS   #1, LIB$SIGNAL                                         ;
                                 50      04 005A8 78$:     MOVL    #4, R0                                                 ; 2675
                                         04 005AB          RET                                                            ;
                                    04   DD 005AC 79$:     PUSHL   #4                                                     ; 2677
                00000000G        00      FB 005AE          CALLS   #1, DBG$GET_MEMORY                                     ;
                                 52      50 005B5          MOVL    R0, PAGE_ENTRY                                         ;
                00000000'        EF      62 0E 005B8       INSQUE  (PAGE_ENTRY), EVENT$PAGE_QUEUE                         ; 2678
                                 50 24   AE D0 005BF       MOVL    PAGE_LIST, R0                                          ; 2680
                              08 A2 04   A0 D0 005C3       MOVL    4(R0), 8(PAGE_ENTRY)                                   ;
                              0C A2 01   B0 005C8          MOVW    #1, 12(PAGE_ENTRY)                                     ; 2681
                              24 AE 24   BE D0 005CC 80$:  MOVL    @PAGE_LIST, PAGE_LIST                                  ; 2683
                                 54 24   AE D0 005D1       MOVL    PAGE_LIST, R4                                          ; 2685
                                 03      54 E9 005D5       BLBC    R4, 81$                                                ;
                                    FF58 31 005D8          BRW     72$                                                    ;
                                 5A   08 AA D0 005DB 81$:  MOVL    8(NOUN_NODE), NOUN_NODE                                ; 2693
                                         52 13 005DF       BEQL    87$                                                    ; 2700
                                         10 DD 005E1       PUSHL   #16                                                    ; 2709
                00000000G        00      FB 005E3          CALLS   #1, DBG$GET_MEMORY                                     ;
                                 56      50 D0 005EA       MOVL    R0, EVENT_ENTRY                                        ;
                66 3C BE      0040 8F   28 005ED          MOVC3   #64, @EVENT_QUEUE, (EVENT_ENTRY)                       ; 2713
                   3C AE           66   0E 005F4          INSQUE  (EVENT_ENTRY), EVENT_QUEUE                             ; 2719
                                 20 A6   D5 005F8          TSTL    32(EVENT_ENTRY)                                        ; 2725
                                         03 13 005FB       BEQL    82$                                                    ;
                                 08 A8   D6 005FD          INCL    8(WHEN_ENTRY)                                          ; 2728
                                 24 A6   D5 00600 82$:     TSTL    36(EVENT_ENTRY)                                        ; 2734
                                         03 13 00603       BEQL    83$                                                    ;
                                 08 A9   D6 00605          INCL    8(DO_ENTRY)                                            ; 2737
                                    FD1B 31 00608 83$:     BRW     54$                                                    ; 2700
                                 01      50 91 0060B 84$:  CMPB    R0, #1                                                 ; 2751
                                         03 13 0060E       BEQL    85$                                                    ;
                                    00A8 31 00610          BRW     99$                                                    ;
                                 50   16 A6 9A 00613 85$:  MOVZBL  22(EVENT_ENTRY), R0                                    ; 2752
                                 05      50 91 00617       CMPB    R0, #5                                                 ; 2754
                                         0A 12 0061A       BNEQ    86$                                                    ;
                              28 A6 00000000' EF 9E 0061C  MOVAB   DBG$OPCODES_CALL, 40(EVENT_ENTRY)                      ; 2755
                                         0D 11 00624       BRB     87$                                                    ;
                                 06      50 91 00626 86$:  CMPB    R0, #6                                                 ; 2756
                                         0A 12 00629       BNEQ    88$                                                    ;
                              28 A6 00000000' EF 9E 0062B  MOVAB   DBG$OPCODES_BRANCH, 40(EVENT_ENTRY)                    ; 2757
                                         7C 11 00633 87$:  BRB     97$                                                    ;
                                 08      50 91 00635 88$:  CMPB    R0, #8                                                 ; 2758
                                         74 13 00638       BEQL    96$                                                    ;
                                 09      50 91 0063A       CMPB    R0, #9                                                 ; 2760
```

```
                                          6A 12 0063D         BNEQ    95$                                        ;
                                          10 DD 0063F         PUSHL   #16                                        ; 2769
                          00000000G  00   01 FB 00641         CALLS   #1, DBG$GET_MEMORY                         ;
                                     57   50 D0 00648         MOVL    R0, OPCODE_LIST                            ;
                                          5A D5 0064B         TSTL    NOUN_NODE                                  ; 2779
                                          2A 12 0064D         BNEQ    90$                                        ;
        0C      04   BC              08   08 ED 0064F         CMPZV   #8, #8, @VERB_NODE, #12                    ; 2780
                                          08 13 00655         BEQL    89$                                        ;
        0B      04   BC              08   08 ED 00657         CMPZV   #8, #8, @VERB_NODE, #11                    ; 2781
                                          1A 12 0065D         BNEQ    90$                                        ;
                       67 00000000'  EF     0040  8F 28 0065F 89$: MOVC3 #64, DBG$OPCODES_STEP_USER, (OPCODE_LIST) ; 2784
           00000000'  EF 00000000'   EF     0040  8F 28 00669      MOVC3 #64, DBG$OPCODES_STEP_USER, -         ; 2785
                                                              DBG$OPCODES_USER                                   ;
                                          14 11 00677         BRB     91$                                        ; 2779
  0040  8F             00            6E   00 2C 00679 90$:    MOVC5   #0, (SP), #0, #64, (OPCODE_LIST)           ; 2789
                                          67    00680                                                            ;
  0040  8F             00            6E   00 2C 00681         MOVC5   #0, (SP), #0, #64, DBG$OPCODES_USER        ; 2790
                                00000000' EF    00688                                                            ;
                                          5A D5 0068D 91$:    TSTL    NOUN_NODE                                  ; 2798
                                          12 13 0068F         BEQL    94$                                        ;
                       00            67   6A E2 00691         BBSS    (NOUN_NODE), (OPCODE_LIST), 92$            ; 2800
                       00 00000000'  EF   6A E2 00695 92$:    BBSS    (NOUN_NODE), DBG$OPCODES_USER, 93$         ; 2801
                                     5A      08  AA D0 0069D 93$: MOVL 8(NOUN_NODE), NOUN_NODE                   ; 2802
                                          EA 11 006A1         BRB     91$                                        ;
                            28  A6        57 D0 006A3 94$:    MOVL    OPCODE_LIST, 40(EVENT_ENTRY)               ; 2808
                                          2C 11 006A7         BRB     101$                                       ; 2752
                                     07   50 91 006A9 95$:    CMPB    R0, #7                                     ; 2812
                                          05 12 006AC         BNEQ    98$                                        ;
                                     28   A6 D4 006AE 96$:    CLRL    40(EVENT_ENTRY)                            ; 2813
                                          22 11 006B1 97$:    BRB     101$                                       ;
                                00000000' EF 9F 006B3 98$:    PUSHAB  P.ABQ                                      ; 2819
                                          0B 11 006B9         BRB     100$                                       ;
                                     02   50 91 006BB 99$:    CMPB    R0, #2                                     ; 2826
                                          15 13 006BE         BEQL    101$                                       ;
                                00000000' EF 9F 006C0         PUSHAB  P.ABR                                      ; 2832
                                          01 DD 006C6 100$:   PUSHL   #1                                         ;
                                00028362  8F DD 006C8         PUSHL   #164706                                    ;
                          00000000G  00   03 FB 006CE         CALLS   #3, LIB$SIGNAL                             ;
                                     03      14  A6 91 006D5 101$: CMPB 20(EVENT_ENTRY), #3                      ; 2839
                                          03 13 006D9         BEQL    102$                                       ;
                                        0128 31 006DB         BRW     112$                                       ;
        0B      04   BC              08   08 ED 006DE 102$:   CMPZV   #8, #8, @VERB_NODE, #11                    ; 2847
                                          0B 12 006E4         BNEQ    103$                                       ;
                                          01 DD 006E6         PUSHL   #1                                         ; 2849
                          00000000G  00   01 FB 006E8         CALLS   #1, DBG$SET_STP_LVL                        ;
                                          34 11 006EF         BRB     105$                                       ;
                                0F 00000000G  00 E9 006F1 103$: BLBC  DBG$GB_EXC_BRE_FLAG, 104$                  ; 2861
                                00028E38  8F DD 006F8         PUSHL   #167480                                    ; 2863
                          00000000G  00   01 FB 006FE         CALLS   #1, LIB$SIGNAL                             ;
                                          1E 11 00705         BRB     105$                                       ;
                                50 00000000G  00 D0 00707 104$: MOVL  DBG$RUNFRAME+64, R0                        ; 2874
                       60            01   00 0C 0070E         PROBER  #0, #1, (R0)                               ;
                                          13 12 00712         BNEQ    106$                                       ;
                                          50 DD 00714         PUSHL   R0                                         ; 2876
                                          01 DD 00716         PUSHL   #1                                         ;
                                000281E0  8F DD 00718         PUSHL   #164320                                    ;
                          00000000G  00   03 FB 0071E         CALLS   #3, LIB$SIGNAL                             ;
```

```
                                                 71  11 00725 105$:   BRB     109$                                                    ;
                                                 02  DD 00727 106$:   PUSHL   #2                                                      ; 2887
                                 00000000G   00  01  FB 00729         CALLS   #1, DBG$SET_STP_LVL                                      ;
                                 00000000G   00      A6  D0 00730         MOVL    28(EVENT_ENTRY), DBG$GL_STEP_NUM                    ; 2892
                                                 5E  13 00738         BEQL    109$                                                     ; 2898
                                             56      66  0F 0073A         REMQUE  (EVENT_ENTRY), EVENT_ENTRY                       ; 2907
                                 00000000'   EF      66  0E 0073D         INSQUE  (EVENT_ENTRY), EVENT$CMD_QUEUE                   ; 2908
                                       08    A6      56  D0 00744         MOVL    EVENT_ENTRY, 8(EVENT_ENTRY)                      ; 2914
                                       0C    A6      56  D0 00748         MOVL    EVENT_ENTRY, 12(EVENT_ENTRY)                     ; 2915
                                             04      A6  91 0074C         CMPB    22(EVENT_ENTRY), #4                              ; 2923
                                                     40  12 00750         BNEQ    108$                                             ;
                                             50  00000000G   00  D0 00752         MOVL    DBG$RUNFRAME+64, R0                      ; 2933
                                             51  00000000G   00  D0 00759         MOVL    DBG$RUNFRAME+56, R1                      ; 2934
                                       20    A6              50  7D 00760         MOVQ    R0, 32(EVENT_ENTRY)                      ; 2933
                                 60          01              00  0C 00764         PROBER  #0, #1, (R0)                             ; 2935
                                                             16  13 00768         BEQL    107$                                     ;
                                             52              60  90 0076A         MOVB    (R0), INSTRUCTION                        ; 2938
                                             04              52  91 0076D         CMPB    INSTRUCTION, #4                          ; 2939
                                                             0E  12 00770         BNEQ    107$                                     ;
                                             50              51  D0 00772         MOVL    R1, STACK_FRAME                          ; 2942
                                 60          14              00  0C 00775         PROBER  #0, #20, (STACK_FRAME)                   ; 2943
                                                             05  13 00779         BEQL    107$                                     ;
                                       24    A6        0C    A0  D0 0077B         MOVL    12(STACK_FRAME), 36(EVENT_ENTRY)         ; 2946
                                                         05  DD 00780 107$:   PUSHL   #5                                       ; 2952
                                 00000000G   00          01  FB 00782         CALLS   #1, DBG$GET_MEMORY                       ;
                                       3C    A6              50  D0 00789         MOVL    CALL_FRAME, 60(EVENT_ENTRY)              ; 2953
                                 60    24    B6              14  28 0078D         MOVC3   #20, @36(EVENT_ENTRY), (CALL_FRAME)      ; 2954
                                       00000000G           00  94 00792 108$:   CLRB    DBG$GB_TAKE_CMD                          ; 2960
                                             57  00000000G   00  D0 00798 109$:   MOVL    DBG$GB_STP_PTR, R7                       ; 2972
                                             67      16      A6  90 0079F         MOVB    22(EVENT_ENTRY), (R7)                    ; 2973
                                             50      18      A6  9E 007A3         MOVAB   24(EVENT_ENTRY), R0                      ; 2975
                                 51          60              01  01  EF 007A7         EXTZV   #1, #1, (R0), R1                         ;
                           01    A7          01              00  51  F0 007AC         INSV    R1, #0, #1, 1(R7)                        ;
                                 51          60              01  03  EF 007B2         EXTZV   #3, #1, (R0), R1                         ; 2977
                                 67          01              09  51  F0 007B7         INSV    R1, #9, #1, (R7)                         ;
                                 51          60              01  05  EF 007BC         EXTZV   #5, #1, (R0), R1                         ; 2979
                                 67          01              0A  51  F0 007C1         INSV    R1, #10, #1, (R7)                        ;
                                 51          60              01  07  EF 007C6         EXTZV   #7, #1, (R0), R1                         ; 2981
                                 67          01              0B  51  F0 007CB         INSV    R1, #11, #1, (R7)                        ;
                                 51          60              01  09  EF 007D0         EXTZV   #9, #1, (R0), R1                         ; 2983
                                 67          01              0C  51  F0 007D5         INSV    R1, #12, #1, (R7)                        ;
                                 0B    04    BC              08  08  ED 007DA         CMPZV   #8, #8, @VERB_NODE, #11                  ; 2989
                                                             21  12 007E0         BNEQ    111$                                     ;
                                                     09      16      A6  91 007E2         CMPB    22(EVENT_ENTRY), #9               ; 2992
                                                             18  12 007E6         BNEQ    110$                                     ;
                                 00000000'   EF  00000000'   EF      0040    8F  28 007E8         MOVC3   #64, DBG$OPCODES_USER, -         ; 2995
                                                                                                     DBG$OPCODES_STEP_USER
                                                         04      A7  00000000'   EF  9E 007F6         MOVAB   DBG$OPCODES_STEP_USER, 4(R7) ; 2996
                                                                 03  11 007FE         BRB     111$                                     ; 2992
                                                         04      A7  D4 00800 110$:   CLRL    4(R7)                                    ; 3000
                                                                 00D5  31 00803 111$:   BRW     128$                                     ; 2989
                                                     50      3C      AE  9E 00806 112$:   MOVAB   EVENT_QUEUE, R0                     ; 3021
                                                     50      3C      AE  D1 0080A         CMPL    EVENT_QUEUE, R0                          ;
                                                                     F3  13 0080E         BEQL    111$                                     ;
                                                     56      3C      AE  D0 00810         MOVL    EVENT_QUEUE, EVENT_ENTRY                 ; 3028
                                                     52  00000000'   EF  D0 00814         MOVL    EVENT$CMD_QUEUE, QUEUE_ENTRY             ; 3033
                                                     54              62  D0 0081B 113$:   MOVL    (QUEUE_ENTRY), NEXTQ_ENTRY               ; 3034
```

```
                50 00000000'   EF   9E 0081E             MOVAB   EVENT$CMD_QUEUE, R0                          ; 3039
                50             52   D1 00825             CMPL    QUEUE_ENTRY, R0
                               03   12 00828             BNEQ    114$
                           0099 31   31 0082A             BRW     127$
                53         14  A2   9E 0082D 114$:       MOVAB   20(QUEUE_ENTRY), R3                         ; 3046
                               63   D5 00831             TSTL    (R3)
                               74   19 00833             BLSS    124$
                50         15  A6   9A 00835             MOVZBL  21(EVENT_ENTRY), R0                         ; 3054
                50         01  A3   91 00839             CMPB    1(R3), R0                                   ; 3055
                               6A   12 0083D             BNEQ    124$
                               50   D5 0083F             TSTL    R0                                          ; 3071
                               35   12 00841             BNEQ    118$
                50         16  A6   9A 00843             MOVZBL  22(EVENT_ENTRY), CODE_1                     ; 3081
                51         02  A3   9A 00847             MOVZBL  2(R3), CODE_2                               ; 3082
                03             50   D1 0084B             CMPL    CODE_1, #3                                  ; 3083
                               0F   13 0084E             BEQL    115$
                04             50   D1 00850             CMPL    CODE_1, #4                                  ; 3084
                               0A   13 00853             BEQL    115$
                03             51   D1 00855             CMPL    CODE_2, #3                                  ; 3085
                               05   13 00858             BEQL    115$
                04             51   D1 0085A             CMPL    CODE_2, #4                                  ; 3086
                               09   12 0085D             BNEQ    117$
             2C A2         2C  A6   D1 0085F 115$:       CMPL    44(EVENT_ENTRY), 44(QUEUE_ENTRY)            ; 3089
                               5A   12 00864 116$:       BNEQ    126$
                               3C   11 00866             BRB     123$
                           34  A2   9F 00868 117$:       PUSHAB  52(QUEUE_ENTRY)                             ; 3092
                           34  A6   9F 0086B             PUSHAB  52(EVENT_ENTRY)                             ; 3091
          0000V CF                 02   FB 0086E             CALLS   #2, COMPARE_VMSDESC                         ; 3092
                4A                 50   E9 00873             BLBC    R0, 126$                                    ; 3095
                                   2C   11 00876             BRB     123$
                01                 50   91 00878 118$:       CMPB    R0, #1                                      ; 3103
                                   22   12 0087B             BNEQ    122$
             04                 05       16  A6   8F 0087D             CASEB   22(EVENT_ENTRY), #5, #4             ; 3104
0011         000A               000A           000A     00882 119$:       .WORD   120$-119$,-
                                                0011     0088A                     120$-119$,-
                                                                                   120$-119$,-
                                                                                   121$-119$,-
                                                                                   121$-119$
                02             A3       16  A6   91 0088C 120$:       CMPB    22(EVENT_ENTRY), 2(R3)      ; 3109
                                   D1   11 00891             BRB     116$
                08             02  A3   91 00893 121$:       CMPB    2(R3), #8                           ; 3114
                               0B   13 00897             BEQL    123$
                09             02  A3   91 00899             CMPB    2(R3), #9                           ; 3116
                               C5   11 0089D             BRB     116$
                02                 50   91 0089F 122$:       CMPB    R0, #2                              ; 3127
                                   07   12 008A2             BNEQ    125$
                03             A3  80   8F 88 008A4 123$:       BISB2   #128, 3(R3)                         ; 3128
                                   15   11 008A9 124$:       BRB     126$
                       00000000'   EF   9F 008AB 125$:       PUSHAB  P.ABS                               ; 3134
                                   01   DD 008B1             PUSHL   #1
                       00028362    8F   DD 008B3             PUSHL   #164706
          00000000G 00                 03   FB 008B9             CALLS   #3, LIB$SIGNAL
                52                 54   D0 008C0 126$:       MOVL    NEXTQ_ENTRY, QUEUE_ENTRY                ; 3145
                               FF55 31   31 008C3             BRW     113$                                    ; 3146
                56                 66   0F 008C6 127$:       REMQUE  (EVENT_ENTRY), EVENT_ENTRY              ; 3153
    00000000'   FF                 66   0E 008C9             INSQUE  (EVENT_ENTRY), @EVENT$CMD_QUEUE+4       ; 3154
          08    A6                 56   D0 008D0             MOVL    EVENT_ENTRY, 8(EVENT_ENTRY)             ; 3160
```

```
                 0C   A6          56  D0 008D4          MOVL    EVENT_ENTRY, 12(EVENT_ENTRY)          ; 3161
                            FF2B  31  008D8          BRW     112$                                  ; 3021
                      56    3C AE   D0  008DB 128$:    MOVL    EVENT_QUEUE, EVENT_ENTRY              ; 3170
                      50    3C AE   9E  008DF 129$:    MOVAB   EVENT_QUEUE, R0                       ; 3171
                      50        56  D1  008E3          CMPL    EVENT_ENTRY, R0                       ;
                                0F  13  008E6          BEQL    130$                                  ;
                      52        66  D0  008E8          MOVL    (EVENT_ENTRY), NEW_EVENT_ENTRY        ; 3175
                                56  DD  008EB          PUSHL   EVENT_ENTRY                           ; 3176
               0000V  CF            01  FB  008ED          CALLS   #1, DELETE_EVENT_ENTRY                ;
                      56            52  D0  008F2          MOVL    NEW_EVENT_ENTRY, EVENT_ENTRY          ; 3177
                                    E8  11  008F5          BRB     129$                                  ; 3171
                      50            01  D0  008F7 130$:    MOVL    #1, R0                                ; 3183
                                        04  008FA          RET                                           ; 3185

; Routine Size:  2299 bytes,    Routine Base:  DBG$CODE + 0AEB
```

```
;  3062     3186  1 GLOBAL ROUTINE DBG$EVENT_SHOW_CANCEL_SYNTAX(INPUT_DESC,
;  3063     3187  1                                                VERB_NODE, MESSAGE_VECT) =
;  3064     3188  1 !
;  3065     3189  1 ! FUNCTION
;  3066     3190  1 !       This routine parses a show or cancel break, trace, or watch point
;  3067     3191  1 !       command,
;  3068     3192  1 !       after the '[SHOW!CANCEL] BREAK', '[SHOW!CANCEL] TRACE', or
;  3069     3193  1 !       '[SHOW!CANCEL] WATCH' has been parsed.  This routine is called
;  3070     3194  1 !       from DBGNSET.
;  3071     3195  1 !
;  3072     3196  1 !       The following syntax is accepted:
;  3073     3197  1 !
;  3074     3198  1 !       <event>
;  3075     3199  1 !
;  3076     3200  1 !       If the event is a watchpoint, <event> is:
;  3077     3201  1 !
;  3078     3202  1 !       /ALL !
;  3079     3203  1 !       [ <address> [ , <address> ]*]
;  3080     3204  1 !
;  3081     3205  1 !       otherwise, <event> is:
;  3082     3206  1 !
;  3083     3207  1 !       /EXCEPTION !
;  3084     3208  1 !       /CALL !
;  3085     3209  1 !       /RETURN !
;  3086     3210  1 !       /BRANCH !
;  3087     3211  1 !       /LINE !
;  3088     3212  1 !       /INSTRUCTION [ = ( <mnemonic> [ , <mnemonic> ] ) ]
;  3089     3213  1 !       /ALL !
;  3090     3214  1 !       [ <address> [ , <address> ]*]
;  3091     3215  1 !
;  3092     3216  1 ! INPUTS
;  3093     3217  1 !       INPUT_DESC:     The current command line input descriptor.
;  3094     3218  1 !       VERB_NODE:      A pointer to a DEBUG verb node, with the COMPOSITE
;  3095     3219  1 !                       attribute set to:
;  3096     3220  1 !
;  3097     3221  1 !                           EVENT$K_SHOW_BREAK, EVENT$K_SHOW_TRACE,
;  3098     3222  1 !                           EVENT$K_SHOW_WATCH,
;  3099     3223  1 !                           EVENT$K_CANCEL_BREAK, EVENT$K_CANCEL_TRACE,
;  3100     3224  1 !                           EVENT$K_CANCEL_WATCH.
;  3101     3225  1 !
;  3102     3226  1 !       MESSAGE_VECT:   The address of a longword to contain the address
;  3103     3227  1 !                       of a message argument vector.
;  3104     3228  1 !
;  3105     3229  1 ! OUTPUTS
;  3106     3230  1 !       The parse tree headed by VERB_NODE is created to represent the
;  3107     3231  1 !       command parsed.  Errors cause this routine tc abort.
;  3108     3232  1 !
;  3109     3233  2   BEGIN
;  3110     3234  2
;  3111     3235  2   MAP
;  3112     3236  2       VERB_NODE :     REF DBG$VERB_NODE;      ! Verb node pointer
;  3113     3237  2
;  3114     3238  2   LOCAL
;  3115     3239  2       STATUS,                                 ! Returned status
;  3116     3240  2       ADVERB_NODE :   REF DBG$ADVERB_NODE,    ! Adverb node pointer
;  3117     3241  2       ADVERB_LINK :   REF DBG$ADVERB_NODE,    ! Adverb node link
;  3118     3242  2       NOUN_NODE :     REF DBG$NOUN_NODE,      ! Noun node pointer
```

```
;  3119     3243  2          NOUN_LINK :     REF DBG$NOUN_NODE;      ! Noun node link
;  3120     3244  2
;  3121     3245  2      BIND
;  3122     3246  2
;  3123     3247  2
;  3124     3248  2          ! Define the character strings to parse.
;  3125     3249  2          !
;  3126     3250  2          DBG$CS_EXCEPTION =      $AC ('EXCEPTION'),
;  3127     3251  2          DBG$CS_INSTRUCTION =    $AC ('INSTRUCTION'),
;  3128     3252  2          DBG$CS_CALL =           $AC ('CALL'),
;  3129     3253  2          DBG$CS_RETURN =         $AC ('RETURN'),
;  3130     3254  2          DBG$CS_BRANCH =         $AC ('BRANCH'),
;  3131     3255  2          DBG$CS_LINE =           $AC ('LINE'),
;  3132     3256  2          DBG$CS_ALL =            $AC ('ALL'),
;  3133     3257  2          DBG$CS_COMMA =          $AC (','),
;  3134     3258  2          DBG$CS_SLASH =          $AC ('/'),
;  3135     3259  2          DBG$CS_CR =             UPLIT BYTE (1, DBG$K_CAR_RETURN);
;  3136     3260  2
;  3137     3261  2
;  3138     3262  2      ! Initialize status to success, in case we don't need to
;  3139     3263  2      ! parse any addresses.
;  3140     3264  2      !
;  3141     3265  2      STATUS = STS$K_SUCCESS;
;  3142     3266  2
;  3143     3267  2
;  3144     3268  2      ! Initialize the verb's adverb and noun pointers to null.
;  3145     3269  2      !
;  3146     3270  2      VERB_NODE [DBG$L_VERB_ADVERB_PTR] = 0;
;  3147     3271  2      VERB_NODE [DBG$L_VERB_OBJECT_PTR] = 0;
;  3148     3272  2
;  3149     3273  2
;  3150     3274  2      ! Initialize the adverb and noun links.
;  3151     3275  2      !
;  3152     3276  2      ADVERB_LINK = VERB_NODE [DBG$L_VERB_ADVERB_PTR];
;  3153     3277  2      NOUN_LINK = VERB_NODE [DBG$L_VERB_OBJECT_PTR];
;  3154     3278  2
;  3155     3279  2
;  3156     3280  2      ! Case on the verb composite - the command type is [SHOW!CANCEL] BREAK,
;  3157     3281  2      ! [SHOW!CANCEL] TRACE, or [SHOW!CANCEL] WATCH.
;  3158     3282  2      !
;  3159     3283  2      SELECTONE .VERB_NODE [DBG$B_VERB_COMPOSITE] OF
;  3160     3284  2          SET
;  3161     3285  2
;  3162     3286  2
;  3163     3287  2          ! [SHOW!CANCEL] BREAK and [SHOW!CANCEL] TRACE come here,
;  3164     3288  2          ! to determine the event type.
;  3165     3289  2          !
;  3166     3290  2          [EVENT$K_SHOW_BREAK, EVENT$K_CANCEL_BREAK,
;  3167     3291  2           EVENT$K_SHOW_TRACE, EVENT$K_CANCEL_TRACE]:
;  3168     3292  3              BEGIN
;  3169     3295  3
;  3170     3294  3
;  3171     3295  3                  ! Parse the command according to the following syntax:
;  3172     3296  3                  !
;  3173     3297  3                  !   /EXCEPTION
;  3174     3298  3                  !   /CALL
;  3175     3299  3                  !   /RETURN
```

```
;  3176          3300  3          !    /BRANCH
;  3177          3301  3          !    /LINE
;  3178          3302  3          !    /INSTRUCTION
;  3179          3303  3          !    /ALL
;  3180          3304  3          !    [ <address> [ , <address> ]*]
;  3181          3305  3          !
;  3182          3306  3          ! First, look for a '/'...
;  3183          3307  3          !
;  3184          3308  4          IF MATCH (DBG$CS_SLASH)
;  3185          3309  3          THEN
;  3186          3310  4              BEGIN
;  3187          3311  4
;  3188          3312  4
;  3189          3313  4              ! Get an node for the qualifier.
;  3190          3314  4              !
;  3191          3315  4              GET_ADVERB_NODE;
;  3192          3316  4
;  3193          3317  4
;  3194          3318  4              ! Parse the qualifier.
;  3195          3319  4              !
;  3196          3320  4              SELECTONE TRUE OF
;  3197          3321  4                  SET
;  3198          3322  4
;  3199          3323  4
;  3200          3324  4                      ! Parse /EXECEPTION.
;  3201          3325  4                      !
;  3202          3326  4                      [MATCH (DBG$CS_EXCEPTION)]:
;  3203          3327  4                          ADVERB_NODE [DBG$B_ADVERB_LITERAL] =
;  3204          3328  4                              ADVERB_EXCEPTION;
;  3205          3329  4
;  3206          3330  4
;  3207          3331  4                      ! Parse /CALL.
;  3208          3332  4                      !
;  3209          3333  4                      [MATCH (DBG$CS_CALL)]:
;  3210          3334  4                          ADVERB_NODE [DBG$B_ADVERB_LITERAL] =
;  3211          3335  4                              ADVERB_CALL;
;  3212          3336  4
;  3213          3337  4
;  3214          3338  4                      ! Parse /BRANCH.
;  3215          3339  4                      !
;  3216          3340  4                      [MATCH (DBG$CS_BRANCH)]:
;  3217          3341  4                          ADVERB_NODE [DBG$B_ADVERB_LITERAL] =
;  3218          3342  4                              ADVERB_BRANCH;
;  3219          3343  4
;  3220          3344  4
;  3221          3345  4                      ! Parse /LINE.
;  3222          3346  4                      !
;  3223          3347  4                      [MATCH (DBG$CS_LINE)]:
;  3224          3348  4                          ADVERB_NODE [DBG$B_ADVERB_LITERAL] =
;  3225          3349  4                              ADVERB_LINE;
;  3226          3350  4
;  3227          3351  4
;  3228          3352  4                      ! Parse /INSTRUCTION
;  3229          3353  4                      !
;  3230          3354  4                      [MATCH (DBG$CS_INSTRUCTION)]:
;  3231          3355  4                          ADVERB_NODE [DBG$B_ADVERB_LITERAL] =
;  3232          3356  4                              ADVERB_INSTRUCTION;
```

```
;  3233          3357  4
;  3234          3358  4
;  3235          3359  4                    ! Parse /ALL
;  3236          3360  4                    !
;  3237          3361  4                    [MATCH (DBG$CS_ALL)]:
;  3238          3362  4                        ADVERB_NODE [DBG$B_ADVERB_LITERAL] =
;  3239          3363  4                            ADVERB_ALL;
;  3240          3364  4
;  3241          3365  4
;  3242          3366  4                    ! Otherwise, we've got a syntax error....
;  3243          3367  4                    !
;  3244          3368  4                    [OTHERWISE]:
;  3245          3369  4                        SYNTAX_ERROR;
;  3246          3370  4                    TES
;  3247          3371  4                END
;  3248          3372  4
;  3249          3373  3            ELSE
;  3250          3374  3
;  3251          3375  3                ! Parse the defaulting <address> [ , <address> ]*.
;  3252          3376  3                !
;  3253          3377  3                BEGIN
;  3254          3378  4
;  3255          3379  4
;  3256          3380  4                ! If we're not standing on a <cr>, get the
;  3257          3381  4                ! <address> [ , <address> ], as long as
;  3258          3382  4                ! an <address> is terminated with a ','.
;  3259          3383  4                !
;  3260          3384  4                IF      (NOT MATCH (DBG$CS_CR))
;  3261          3385  5                THEN
;  3262          3386  4                    DO
;  3263          3387  4                        BEGIN
;  3264          3388  5
;  3265          3389  5                    ! Get a noun node for an <address>.
;  3266          3390  5                    !
;  3267          3391  5                    GET_NOUN_NODE;
;  3268          3392  5
;  3269          3393  5
;  3270          3394  5                    ! Get an <address> - the is a status warning
;  3271          3395  5                    ! if more is on the line.
;  3272          3396  5                    !
;  3273          3397  5                    STATUS = DBG$NPARSE_ADDRESS(.INPUT_DESC,
;  3274          3398  5                                NOUN_NODE[DBG$L_NOUN_VALUE],
;  3275          3399  5                                .DBG$GB_RADIX[DBG$B_RADIX_INPUT],
;  3276          3400  5                                TOKEN$K_TERM_COMMA, .MESSAGE_VEC);
;  3277          3401  5
;  3278          3402  5
;  3279          3403  5                    ! If the status indicates success, there is no
;  3280          3404  5                    ! more on the line, and we exit this loop.
;  3281          3405  5                    !
;  3282          3406  5                    IF .STATUS EQLU STS$K_SUCCESS
;  3283          3407  5                    THEN
;  3284          3408  5                        EXITLOOP;
;  3285          3409  5
;  3286          3410  5
;  3287          3411  5                    ! If the status is not a warning, something is
;  3288          3412  5                    ! wrong, so return with a failure.
;  3289          3413  5
```

```
;  3290     3414  5                    !
;  3291     3415  5                    IF .STATUS NEQU STS$K_WARNING
;  3292     3416  5                    THEN
;  3293     3417  5                        RETURN .STATUS
;  3294     3418  5                    END
;  3295     3419  4                UNTIL (MATCH (DBG$CS_COMMA));
;  3296     3420  4
;  3297     3421  3            END;
;  3298     3422  3
;  3299     3423  2        END;
;  3300     3424  2
;  3301     3425  2
;  3302     3426  2        ! Parse [SHOW:CANCEL] WATCH [[ /ALL : <address> [ , <address> ]*]].
;  3303     3427  2        !
;  3304     3428  2        [EVENT$K_SHOW_WATCH, EVENT$K_CANCEL_WATCH]:
;  3305     3429  3            BEGIN
;  3306     3430  4            IF MATCH (DBG$CS_SLASH)
;  3307     3431  3            THEN
;  3308     3432  4                BEGIN
;  3309     3433  4                GET_ADVERB_NODE;
;  3310     3434  4                NEED_MATCH (DBG$CS_ALL);
;  3311     3435  4                ADVERB_NODE [DBG$B_ADVERB_LITERAL] = ADVERB_ALL;
;  3312     3436  4                END
;  3313     3437  4
;  3314     3438  3            ELSE
;  3315     3439  4                BEGIN
;  3316     3440  5                IF NOT MATCH (DBG$CS_CR)
;  3317     3441  4                THEN
;  3318     3442  4                    ! Get the <address> [ , <address> ], as long as
;  3319     3443  4                    ! an <address> is terminated with a ','.
;  3320     3444  4                    !
;  3321     3445  4                    DO
;  3322     3446  5                        BEGIN
;  3323     3447  5
;  3324     3448  5                        ! Get a noun node for an <address>.
;  3325     3449  5                        !
;  3326     3450  5                        GET_NOUN_NODE;
;  3327     3451  5
;  3328     3452  5                        ! Get an <address> - the is a status warning
;  3329     3453  5                        ! if more is on the line.
;  3330     3454  5                        !
;  3331     3455  5                        STATUS = DBG$NPARSE_ADDRESS(.INPUT_DESC,
;  3332     3456  5                                    NOUN_NODE[DBG$L_NOUN_VALUE],
;  3333     3457  5                                    .DBG$GB_RADIX[DBG$B_RADIX_INPUT],
;  3334     3458  5                                    TOKEN$K_TERM_COMMA, .MESSAGE_VECT);
;  3335     3459  5
;  3336     3460  5                        ! If the status indicates success, there is no
;  3337     3461  5                        ! more on the line, and we exit this loop.
;  3338     3462  5                        !
;  3339     3463  5                        IF .STATUS EQLU STS$K_SUCCESS
;  3340     3464  5                        THEN
;  3341     3465  5                            EXITLOOP;
;  3342     3466  5
;  3343     3467  5
;  3344     3468  5
;  3345     3469  5
;  3346     3470  5
```

```
;  3347     3471  5                    ! If the status is not a warning, something is
;  3348     3472                       ! wrong, so return with a failure.
;  3349     3473                       !
;  3350     3474                       IF .STATUS NEQU STS$K_WARNING
;  3351     3475                       THEN
;  3352     3476                           RETURN .STATUS;
;  3353     3477
;  3354     3478                       END
;  3355     3479                   WHILE (MATCH (DBG$CS_COMMA));
;  3356     3480
;  3357     3481               END;
;  3358     3482
;  3359     3483           END;
;  3360     3484
;  3361     3485       TES;
;  3362     3486
;  3363     3487
;  3364     3488   ! Check the status after parsing <address> [ , <address> ] or <mnemonic>
;  3365     3489   ! [ , <mnemonic> ].  If there's nothing left on the line, the status will
;  3366     3490   ! be success, and we can leave.  Otherwise we signal a syntax error.
;  3367     3491   !
;  3368     3492   IF .STATUS EQLU STS$K_SUCCESS THEN RETURN STS$K_SUCCESS;
;  3369     3493   .MESSAGE_VECT = DBG$NSYNTAX_ERROR (DBG$NNEXT_WORD (.INPUT_DESC));
;  3370     3494   RETURN STS$K_SEVERE
;  3371     3495   !
;  3372     3496  1    END;


                                                        .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                    4E  4F  49  54  50  45  43  58  45  09  001A1 P.ABT:   .ASCII  <9>\EXCEPTION\                    ;
            4E  4F  49  54  43  55  52  54  53  4E  49  0B  001AB P.ABU:   .ASCII  <11>\INSTRUCTION\                ;
                                        4C  4C  41  43  04  001B7 P.ABV:   .ASCII  <4>\CALL\                       ;
                                    4E  52  55  54  45  52  06  001BC P.ABW:   .ASCII  <6>\RETURN\                   ;
                                    48  43  4E  41  52  42  06  001C3 P.ABX:   .ASCII  <6>\BRANCH\                   ;
                                            45  4E  49  4C  04  001CA P.ABY:   .ASCII  <4>\LINE\                     ;
                                                4C  4C  41  03  001CF P.ABZ:   .ASCII  <3>\ALL\                     ;
                                                        2C  01  001D3 P.ACA:   .ASCII  <1>\,\                       ;
                                                        2F  01  001D5 P.ACB:   .ASCII  <1>\/\                       ;
                                                        0D  01  001D7 P.ACC:   .BYTE   1, 13                        ;

                                                DBG$CS_EXCEPTION=   P.ABT
                                                DBG$CS_INSTRUCTION= P.ABU
                                                DBG$CS_CALL=        P.ABV
                                                DBG$CS_RETURN=      P.ABW
                                                DBG$CS_BRANCH=      P.ABX
                                                DBG$CS_LINE=        P.ABY
                                                DBG$CS_ALL=         P.ABZ
                                                DBG$CS_COMMA=       P.ACA
                                                DBG$CS_SLASH=       P.ACB
                                                DBG$CS_CR=          P.ACC

                                                        .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                        0FFC 00000      .ENTRY  DBG$EVENT_SHOW_CANCEL_SYNTAX, Save R2,R3,-  ; 3186
```

```
                                                        R4,R5,R6,R7,R8,R9,R10,R11
5B 00000000G   00  9E 00002         MOVAB   DBG$GB_RADIX, R11
5A 00000000G   00  9E 00009         MOVAB   DBG$GET_TEMPMEM, R10
59 00000000'   EF  9E 00010         MOVAB   DBG$CS_CR, R9
58 00000000G   00  9E 00017         MOVAB   DBG$NMATCH, R8
57             01  D0 0001E         MOVL    #1, STATUS                              ;3265
50       08    AC  D0 00021         MOVL    VERB_NODE, R0                           ;3270
         04    A0  7C 00025         CLRQ    4(R0)
55       04    A0  9E 00028         MOVAB   4(R0), ADVERB_LINK                      ;3276
56       08    A0  9E 0002C         MOVAB   8(R0), NOUN_LINK                        ;3277
50       01    A0  9A 00030         MOVZBL  1(R0), R0                               ;3283
                 05  15 00034         BLEQ    1$                                      ;3290
02               50  91 00036         CMPB    R0, #2
                 0D  1B 00039         BLEQU   2$
09               50  91 0003B 1$:     CMPB    R0, #9
                 08  13 0003E         BEQL    2$
0B               50  91 00040         CMPB    R0, #11
                 03  13 00043         BEQL    2$
              00EF  31 00045         BRW     14$
                 01  DD 00048 2$:     PUSHL   #1                                      ;3308
           FE    A9  9F 0004A         PUSHAB  DBG$CS_SLASH
54       04    AC  D0 0004D         MOVL    INPUT_DESC, R4
                 54  DD 00051         PUSHL   R4
68               03  FB 00053         CALLS   #3, DBG$NMATCH
03               50  E8 00056         BLBS    R0, 3$
              008C  31 00059         BRW     11$
                 03  DD 0005C 3$:     PUSHL   #3                                      ;3310
6A               01  FB 0005E         CALLS   #1, DBG$GET_TEMPMEM
52               50  D0 00061         MOVL    R0, ADVERB_NODE
         08    A2  D4 00064         CLRL    8(ADVERB_NODE)
65               52  D0 00067         MOVL    ADVERB_NODE, (ADVERB_LINK)
55       08    A2  9E 0006A         MOVAB   8(ADVERB_NODE), ADVERB_LINK
                 01  DD 0006E         PUSHL   #1                                      ;3326
           CA    A9  9F 00070         PUSHAB  DBG$CS_EXCEPTION
                 54  DD 00073         PUSHL   R4
68               03  FB 00075         CALLS   #3, DBG$NMATCH
01               50  D1 00078         CMPL    R0, #1
                 05  12 0007B         BNEQ    4$
62               0B  90 0007D         MOVB    #11, (ADVERB_NODE)                      ;3327
                 4E  11 00080         BRB     8$
                 01  DD 00082 4$:     PUSHL   #1                                      ;3333
           E0    A9  9F 00084         PUSHAB  DBG$CS_CALL
                 54  DD 00087         PUSHL   R4
68               03  FB 00089         CALLS   #3, DBG$NMATCH
01               50  D1 0008C         CMPL    R0, #1
                 05  12 0008F         BNEQ    5$
62               12  90 00091         MOVB    #18, (ADVERB_NODE)                      ;3334
                 3A  11 00094         BRB     8$
                 01  DD 00096 5$:     PUSHL   #1                                      ;3340
           EC    A9  9F 00098         PUSHAB  DBG$CS_BRANCH
                 54  DD 0009B         PUSHL   R4
68               03  FB 0009D         CALLS   #3, DBG$NMATCH
01               50  D1 000A0         CMPL    R0, #1
                 05  12 000A3         BNEQ    6$
62               13  90 000A5         MOVB    #19, (ADVERB_NODE)                      ;3341
                 26  11 000A8         BRB     8$
                 01  DD 000AA 6$:     PUSHL   #1                                      ;3347
```

```
                F3    A9  9F 000AC          PUSHAB  DBG$CS_LINE                                    ;
                      54  DD 000AF          PUSHL   R4                                             ;
                68    03  FB 000B1          CALLS   #3, DBG$NMATCH                                 ;
                01    50  D1 000B4          CMPL    R0, #1                                         ;
                      05  12 000B7          BNEQ    7$                                             ;
                62    14  90 000B9          MOVB    #20, (ADVERB_NODE)                             ; 3348
                      12  11 000BC          BRB     8$                                             ;
                      01  DD 000BE 7$:       PUSHL   #1                                             ; 3354
                D4    A9  9F 000C0          PUSHAB  DBG$CS_INSTRUCTION                             ;
                      54  DD 000C3          PUSHL   R4                                             ;
                68    03  FB 000C5          CALLS   #3, DBG$NMATCH                                 ;
                01    50  D1 000C8          CMPL    R0, #1                                         ;
                      06  12 000CB          BNEQ    9$                                             ;
                62    15  90 000CD          MOVB    #21, (ADVERB_NODE)                             ; 3355
                    010B  31 000D0 8$:       BRW     22$                                            ;
                      01  DD 000D3 9$:       PUSHL   #1                                             ; 3361
                F8    A9  9F 000D5          PUSHAB  DBG$CS_ALL                                     ;
                      54  DD 000D8          PUSHL   R4                                             ;
                68    03  FB 000DA          CALLS   #3, DBG$NMATCH                                 ;
                01    50  D1 000DD          CMPL    R0, #1                                         ;
                      03  13 000E0          BEQL    10$                                            ;
                    0087  31 000E2          BRW     15$                                            ;
                    00A3  31 000E5 10$:      BRW     17$                                            ;
                      01  DD 000E8 11$:      PUSHL   #1                                             ; 3385
              0210    8F  BB 000EA          PUSHR   #^M<R4,R9>                                     ;
                68    03  FB 000EE          CALLS   #3, DBG$NMATCH                                 ;
                DC    50  E8 000F1          BLBS    R0, 8$                                         ;
                      04  DD 000F4 12$:      PUSHL   #4                                             ; 3388
                6A    01  FB 000F6          CALLS   #1, DBG$GET_TEMPMEM                            ;
                53    50  D0 000F9          MOVL    R0, NOUN_NODE                                  ;
              08      A3  D4 000FC          CLRL    8(NOUN_NODE)                                   ;
                66    53  D0 000FF          MOVL    NOUN_NODE, (NOUN_LINK)                         ;
                56    08  A3 9E 00102       MOVAB   8(NOUN_NODE), NOUN_LINK                        ;
                      0C  AC DD 00106       PUSHL   MESSAGE_VECT                                   ; 3401
                      01  DD 00109          PUSHL   #1                                             ; 3399
                7E        6B 9A 0010B       MOVZBL  DBG$GB_RADIX, -(SP)                            ; 3400
                          53 DD 0010E       PUSHL   NOUN_NODE                                      ; 3399
                          54 DD 00110       PUSHL   R4                                             ;
00000000G       00        05 FB 00112       CALLS   #5, DBG$NPARSE_ADDRESS                         ;
                57        50 D0 00119       MOVL    R0, STATUS                                     ;
                01        57 D1 0011C       CMPL    STATUS, #1                                     ; 3407
                          AF 13 0011F       BEQL    8$                                             ;
                          57 D5 00121       TSTL    STATUS                                         ; 3415
                          03 13 00123       BEQL    13$                                            ;
                        00A5 31 00125       BRW     20$                                            ;
                          01 DD 00128 13$:    PUSHL   #1                                             ; 3419
                      FC  A9 9F 0012A       PUSHAB  DBG$CS_COMMA                                   ;
                          54 DD 0012D       PUSHL   R4                                             ;
                68        03 FB 0012F       CALLS   #3, DBG$NMATCH                                 ;
                9B        50 E9 00132       BLBC    R0, 8$                                         ;
                          BD 11 00135       BRB     12$                                            ;
                0E        50 91 00137 14$:    CMPB    R0, #14                                        ; 3428
                          94 12 0013A       BNEQ    8$                                             ;
                          01 DD 0013C       PUSHL   #1                                             ; 3430
                      FE  A9 9F 0013E       PUSHAB  DBG$CS_SLASH                                   ;
                54    04  AC D0 00141       MOVL    INPUT_DESC, R4                                 ;
                          54 DD 00145       PUSHL   R4                                             ;
```

```
                68              03  FB 00147           CALLS   #3, DBG$NMATCH                                  ;
                43              50  E9 0014A           BLBC    R0, 18$                                         ;
                                03  DD 0014D           PUSHL   #3                                              ; 3432
                6A              01  FB 0014F           CALLS   #1, DBG$GET_TEMPMEM                             ;
                52              50  D0 00152           MOVL    R0, ADVERB_NODE                                 ;
                          08    A2  D4 00155           CLRL    8(ADVERB_NODE)                                  ;
                65              52  D0 00158           MOVL    ADVERB_NODE, (ADVERB_LINK)                      ;
                55        08    A2  9E 0015B           MOVAB   8(ADVERB_NODE), ADVERB_LINK                     ;
                                01  DD 0015F           PUSHL   #1                                              ; 3434
                          FB    A9  9F 00161           PUSHAB  DBG$CS_ALL                                      ;
                                54  DD 00164           PUSHL   R4                                              ;
                68              03  FB 00166           CALLS   #3, DBG$NMATCH                                  ;
                1F              50  E8 00169           BLBS    R0, 17$                                         ;
                                01  DD 0016C 15$:      PUSHL   #1                                              ;
                        0210    8F  BB 0016E           PUSHR   #^M<R4,R9>                                      ;
                68              03  FB 00172           CALLS   #3, DBG$NMATCH                                  ;
                0F              50  E9 00175           BLBC    R0, 16$                                         ;
                    000280D0    8F  DD 00178           PUSHL   #164048                                         ;
00000000G       00              01  FB 0017E           CALLS   #1, DBG$NMAKE_ARG_VECT                          ;
                                73  11 00185           BRB     25$                                             ;
                                54  DD 00187 16$:      PUSHL   R4                                              ;
                                5F  11 00189           BRB     24$                                             ;
                62              17  90 0018B 17$:      MOVB    #23, (ADVERB_NODE)                              ; 3435
                                4E  11 0018E           BRB     22$                                             ; 3430
                                01  DD 00190 18$:      PUSHL   #1                                              ; 3440
                        0210    8F  BB 00192           PUSHR   #^M<R4,R9>                                      ;
                68              03  FB 00196           CALLS   #3, DBG$NMATCH                                  ;
                42              50  E8 00199           BLBS    R0, 22$                                         ;
                                04  DD 0019C 19$:      PUSHL   #4                                              ; 3446
                6A              01  FB 0019E           CALLS   #1, DBG$GET_TEMPMEM                             ;
                53              50  D0 001A1           MOVL    R0, NOUN_NODE                                   ;
                          08    A3  D4 001A4           CLRL    8(NOUN_NODE)                                    ;
                66              53  D0 001A7           MOVL    NOUN_NODE, (NOUN_LINK)                          ;
                56        08    A3  9E 001AA           MOVAB   8(NOUN_NODE), NOUN_LINK                         ;
                          0C    AC  DD 001AE           PUSHL   MESSAGE_VECT                                    ; 3460
                                01  DD 001B1           PUSHL   #1                                              ; 3458
                7E              6B  9A 001B3           MOVZBL  DBG$GB_RADIX, -(SP)                             ; 3459
                                53  DD 001B6           PUSHL   NOUN_NODE                                       ; 3458
                                54  DD 001B8           PUSHL   R4                                              ;
00000000G       00              05  FB 001BA           CALLS   #5, DBG$NPARSE_ADDRESS                          ;
                57              50  D0 001C1           MOVL    R0, STATUS                                      ;
                01              57  D1 001C4           CMPL    STATUS, #1                                      ; 3466
                                15  13 001C7           BEQL    22$                                             ;
                                57  D5 001C9           TSTL    STATUS                                          ; 3474
                                04  13 001CB           BEQL    21$                                             ;
                50              57  D0 001CD 20$:      MOVL    STATUS, R0                                      ; 3476
                                04  001D0             RET                                                     ;
                                01  DD 001D1 21$:      PUSHL   #1                                              ; 3479
                          FC    A9  9F 001D3           PUSHAB  DBG$CS_COMMA                                    ;
                                54  DD 001D6           PUSHL   R4                                              ;
                68              03  FB 001D8           CALLS   #3, DBG$NMATCH                                  ;
                BE              50  E8 001DB           BLBS    R0, 19$                                         ;
                01              57  D1 001DE 22$:      CMPL    STATUS, #1                                      ; 3492
                                04  12 001E1           BNEQ    23$                                             ;
                50              01  D0 001E3           MOVL    #1, R0                                          ;
                                04  001E6             RET                                                     ;
                          04    AC  DD 001E7 23$:      PUSHL   INPUT_DESC                                      ; 3493
```

```
                00000000G  00        01 FB 001EA 24$:    CALLS   #1, DBG$NNEXT_WORD                    ;
                                     50 DD 001F1         PUSHL   R0                                   ;
                00000000G  00        01 FB 001F3         CALLS   #1, DBG$NSYNTAX_ERROR                ;
                      0C   BC        50 D0 001FA 25$:     MOVL    R0, @MESSAGE_VECT                   ;
                           50        04 D0 001FE         MOVL    #4, R0                              ; 3494
                                        04 00201         RET                                         ; 3496

; Routine Size:  514 bytes,    Routine Base:  DBG$CODE + 13E6
```

```
;  3374          3497  1 GLOBAL ROUTINE DBG$EVENT_SHOW_CANCEL_SEMANTICS(VERB_NODE, MESSAGE_VECT) =
;  3375          3498  1 !
;  3376          3499  1 ! FUNCTION
;  3377          3500  1 !       This routine builds the event data stuctures based on the
;  3378          3501  1 !       parse tree passed via the VERB_NODE address.  This routine
;  3379          3502  1 !       is called from DBGNSET.
;  3380          3503  1 !
;  3381          3504  1 ! INPUTS
;  3382          3505  1 !       VERB_NODE:      A pointer to a DEBUG verb node, which defines,
;  3383          3506  1 !                       along with the verb's descendants, the desired
;  3384          3507  1 !                       command.
;  3385          3508  1 !       MESSAGE_VECT:   The address of a longword to contain the address
;  3386          3509  1 !                       of a message argument vector.
;  3387          3510  1 !
;  3388          3511  1 ! OUTPUTS
;  3389          3512  1 !       The event data structure(s) are updated to include the new event,
;  3390          3513  1 !       and may include:
;  3391          3514  1 !
;  3392          3515  1 !           EVENT$CMD_QUEUE,
;  3393          3516  1 !           EVENT$WHEN_QUEUE,
;  3394          3517  1 !           EVENT$DO_LIST_QUEUE
;  3395          3518  1 !
;  3396          3519  1
;  3397          3520  2     BEGIN
;  3398          3521  2
;  3399          3522  2     MAP
;  3400          3523  2         VERB_NODE: REF DBG$VERB_NODE;   ! Verb node pointer
;  3401          3524  2
;  3402          3525  2     MACRO
;  3403        M 3526  2         SHOW_OR_DELETE_EVENT_ENTRY (EVENT_ENTRY) =
;  3404        M 3527  2             BEGIN
;  3405        M 3528  2             IF .VERB_NODE [DBG$B_VERB_LITERAL] EQLU DBG$K_SHOW_VERB
;  3406        M 3529  2             THEN
;  3407        M 3530  2                 SHOW_EVENT_ENTRY (.EVENT_ENTRY)
;  3408        M 3531  2
;  3409        M 3532  2             ELSE
;  3410        M 3533  2                 EVENT_ENTRY [EVENT$V_DELETED] = TRUE;
;  3411        M 3534  2
;  3412        M 3535  2             SHOWED_CANCELED_ONE = TRUE;
;  3413        M 3536  2             END
;  3414          3537  2         %;
;  3415          3538  2
;  3416          3539  2     LOCAL
;  3417          3540  2         DUMMY,
;  3418          3541  2         EVENT_QUEUE: QUEUE_HEAD,                    ! Working queue head
;  3419          3542  2         QUEUE_ENTRY: REF EVENT$EVENT_DESCRIPTOR,    ! Event queue entry pointer
;  3420          3543  2         NEXTQ_ENTRY: REF EVENT$EVENT_DESCRIPTOR,    ! Event queue entry pointer
;  3421          3544  2         EVENT_ENTRY: REF EVENT$EVENT_DESCRIPTOR,    ! Event entry pointer
;  3422          3545  2         DO_ENTRY: REF EVENT$DO_LIST_DESCRIPTOR,     ! DO List entry pointer
;  3423          3546  2         WHEN_ENTRY: REF EVENT$WHEN_DESCRIPTOR,      ! WHEN entry pointer
;  3424          3547  2         ADVERB_NODE: REF DBG$ADVERB_NODE,           ! Adverb node link
;  3425          3548  2         NOUN_NODE: REF DBG$NOUN_NODE,               ! Noun node link
;  3426          3549  2         PRIMARY: REF DBG$PRIMARY,                   ! Event primary descriptor
;  3427          3550  2         VALUE: REF DBG$VALDESC,                     ! Event value descriptor
;  3428          3551  2         ADDRESS: VECTOR [2],                        ! Event address
;  3429          3552  2         ADR_TYP,                                    ! Event address type
;  3430          3553  2         SHOW_CANCEL_ALL,                            ! Show or Cancel all ?
```

```
;  3431     3554  2          SHOWED_CANCELED_ANY,                    ! Did we show/cancel any yet ?
;  3432     3555  2          SHOWED_CANCELED_ONE,                    ! Did we show/cancel one yet ?
;  3433     3556  2          COUNTER;                                ! Counter variable
;  3434     3557  2
;  3435     3558  2
;  3436     3559  2
;  3437     3560  2      ! Initialize the working queue to be empty.
;  3438     3561  2      !
;  3439     3562  2      EVENT_QUEUE [L_QUEUE_FLINK] = EVENT_QUEUE;
;  3440     3563  2      EVENT_QUEUE [L_QUEUE_BLINK] = EVENT_QUEUE;
;  3441     3564  2
;  3442     3565  2
;  3443     3566  2      ! Get an event descriptor from permanent memory, and
;  3444     3567  2      ! link it into the working queue.
;  3445     3568  2      !
;  3446     3569  2      EVENT_ENTRY = DBG$GET_MEMORY (EVENT$K_EVENT_DESCRIPTOR_SIZE);
;  3447     3570  2      INSQUE (.EVENT_ENTRY, EVENT_QUEUE);
;  3448     3571  2
;  3449     3572  2
;  3450     3573  2      ! Initialize the command kind to access (default).
;  3451     3574  2      !
;  3452     3575  2      EVENT_ENTRY [EVENT$B_CMD_KIND] = EVENT$K_KIND_ACC;
;  3453     3576  2
;  3454     3577  2
;  3455     3578  2      ! Set the event's command type, found in the verb.
;  3456     3579  2      !
;  3457     3580  2      SELECTONE .VERB_NODE [DBG$B_VERB_COMPOSITE] OF
;  3458     3581  2          SET
;  3459     3582  2
;  3460     3583  2          [EVENT$K_SHOW_BREAK, EVENT$K_CANCEL_BREAK]:
;  3461     3584  2              EVENT_ENTRY [EVENT$B_CMD_TYPE] = EVENT$K_TYPE_BREAK;
;  3462     3585  2
;  3463     3586  2          [EVENT$K_CANCEL_BREAK_EXC]:
;  3464     3587  3              BEGIN
;  3465     3588  3              EVENT_ENTRY [EVENT$B_CMD_TYPE] = EVENT$K_TYPE_BREAK;
;  3466     3589  3              EVENT_ENTRY [EVENT$B_CMD_KIND] = EVENT$K_KIND_EXC;
;  3467     3590  2              END;
;  3468     3591  2
;  3469     3592  2          [EVENT$K_SHOW_TRACE, EVENT$K_CANCEL_TRACE]:
;  3470     3593  2              EVENT_ENTRY [EVENT$B_CMD_TYPE] = EVENT$K_TYPE_TRACE;
;  3471     3594  2
;  3472     3595  2          [EVENT$K_SHOW_WATCH, EVENT$K_CANCEL_WATCH]:
;  3473     3596  2              EVENT_ENTRY [EVENT$B_CMD_TYPE] = EVENT$K_TYPE_WATCH;
;  3474     3597  2
;  3475     3598  2          [OTHERWISE]:
;  3476     3599  2              $DBG_ERROR('DBGEVENT\EVENT_SHOW_CANCEL_SEMANTICS 10');
;  3477     3600  2
;  3478     3601  2          TES;
;  3479     3602  2
;  3480     3603  2
;  3481     3604  2      ! Initialize Has Value Descriptor flag.
;  3482     3605  2      !
;  3483     3606  2      EVENT_ENTRY [EVENT$V_HAS_VAL_DSCR] = FALSE;
;  3484     3607  2
;  3485     3608  2
;  3486     3609  2      ! Initialize the Primary pointer to NULL.
;  3487     3610  2      !
```

```
;  3488          3611  2     EVENT_ENTRY [EVENT$L_PRIMARY] = 0;
;  3489          3612  2
;  3490          3613  2
;  3491          3614  2     ! Point to the verb's noun list.
;  3492          3615  2     !
;  3493          3616  2     NOUN_NODE = .VERB_NODE [DBG$L_VERB_OBJECT_PTR];
;  3494          3617  2
;  3495          3618  2
;  3496          3619  2     ! Point to the verb's adverb list.
;  3497          3620  2     !
;  3498          3621  2     ADVERB_NODE = .VERB_NODE [DBG$L_VERB_ADVERB_PTR];
;  3499          3622  2
;  3500          3623  2
;  3501          3624  2     ! Initialize the All flag.
;  3502          3625  2     !
;  3503          3626  2     SHOW_CANCEL_ALL = FALSE;
;  3504          3627  2
;  3505          3628  2
;  3506          3629  2     ! If an adverb exists, process it.
;  3507          3630  2     !
;  3508          3631  2     IF .ADVERB_NODE NEQA 0
;  3509          3632  2     THEN
;  3510          3633  3         BEGIN
;  3511          3634  3
;  3512          3635  3
;  3513          3636  3         ! Process adverbs based on the literal set by the
;  3514          3637  3         ! syntactic processing.
;  3515          3638  3         !
;  3516          3639  3         SELECTONE .ADVERB_NODE [DBG$B_ADVERB_LITERAL] OF
;  3517          3640  3             SET
;  3518          3641  3
;  3519          3642  3
;  3520          3643  3             ! Process the /EXCEPTION adverb.
;  3521          3644  3             !
;  3522          3645  3             [ADVERB_EXCEPTION]:
;  3523          3646  3                     EVENT_ENTRY [EVENT$B_CMD_KIND] = EVENT$K_KIND_EXC;
;  3524          3647  3
;  3525          3648  3
;  3526          3649  3             ! Process the /CALL, /RETURN, /BRANCH, /LINE, /OPCODE
;  3527          3650  3             ! and /INSTRUCTION instruction-kind adverbs.
;  3528          3651  3             !
;  3529          3652  3             [ADVERB_CALL, ADVERB_RETURN, ADVERB_BRANCH,
;  3530          3653  3              ADVERB_LINE, ADVERB_OPCODE, ADVERB_INSTRUCTION]:
;  3531          3654  4                 BEGIN
;  3532          3655  4
;  3533          3656  4
;  3534          3657  4                 ! Define the event kind to be instruction.
;  3535          3658  4                 !
;  3536          3659  4                 EVENT_ENTRY [EVENT$B_CMD_KIND] = EVENT$K_KIND_INS;
;  3537          3660  4
;  3538          3661  4
;  3539          3662  4                 ! Define the subkind according to the adverb.
;  3540          3663  4                 !
;  3541          3664  4                 EVENT_ENTRY [EVENT$B_SUB_KIND] =
;  3542          3665  5                     (SELECTONE .ADVERB_NODE [DBG$B_ADVERB_LITERAL] OF
;  3543          3666  5                         SET
;  3544          3667  5                         [ADVERB_CALL]:
```

```
;  3545     3668  5                        EVENT$K_INS_CALL;
;  3546     3669  5                    [ADVERB_BRANCH]:
;  3547     3670  5                        EVENT$K_INS_BRAN;
;  3548     3671  5                    [ADVERB_LINE]:
;  3549     3672  5                        EVENT$K_INS_LINE;
;  3550     3673  5                    [ADVERB_OPCODE]:
;  3551     3674  5                        EVENT$K_INS_USER;
;  3552     3675  5                    [ADVERB_INSTRUCTION]:
;  3553     3676  5                        EVENT$K_INS_EVRY;
;  3554     3677  5                    [OTHERWISE]:
;  3555     3678  5                        $DBG_ERROR('DBGEVENT\EVENT_SHOW_CANCEL_SEMANTICS 20');
;  3556     3679  5                    TES
;  3557     3680  5                )
;  3558     3681  5                END;
;  3559     3682  5
;  3560     3683  5
;  3561     3684  5                ! Process the /ALL adverb.
;  3562     3685  5                !
;  3563     3686  5                [ADVERB_ALL]:
;  3564     3687  5                    SHOW_CANCEL_ALL = TRUE;
;  3565     3688  5
;  3566     3689  5
;  3567     3690  5                ! An invalid adverb literal - yell !
;  3568     3691  5                !
;  3569     3692  5                [OTHERWISE]:
;  3570     3693  5                    $DBG_ERROR('DBGEVENT\EVENT_SHOW_CANCEL_SEMANTICS 30');
;  3571     3694  5
;  3572     3695  5                TES;
;  3573     3696  5
;  3574     3697  2            END;
;  3575     3698  2
;  3576     3699  2
;  3577     3700  2            ! If there are no noun nodes, set the All flag if this is a
;  3578     3701  2            ! SHOW command.  Otherwise, make sure the command kind is
;  3579     3702  2            ! (still) access, and process the address(es).
;  3580     3703  2            !
;  3581     3704  2            IF .NOUN_NODE EQLA 0
;  3582     3705  2            THEN
;  3583     3706  3                BEGIN
;  3584     3707  3                IF .VERB_NODE [DBG$B_VERB_LITERAL] EQLU DBG$K_SHOW_VERB
;  3585     3708  3                THEN
;  3586     3709  3                    SHOW_CANCEL_ALL = TRUE;
;  3587     3710  3
;  3588     3711  3                END
;  3589     3712  3
;  3590     3713  2            ELSE
;  3591     3714  3                BEGIN
;  3592     3715  3
;  3593     3716  3
;  3594     3717  3                ! Build duplicate event entries, one for each
;  3595     3718  3                ! address found in the noun node list.
;  3596     3719  3                !
;  3597     3720  3                WHILE TRUE DO
;  3598     3721  4                    BEGIN
;  3599     3722  4
;  3600     3723  4
;  3601     3724  4                    ! Using the address's primary descriptor, get the address.
```

```
;  3602        3725  4              !
;  3603        3726  4              PRIMARY = .NOUN_NODE [DBG$L_NOUN_VALUE];
;  3604        3727  4              EVENT_ENTRY [EVENT$V_BREAK_ADDRESS] = FALSE;
;  3605        3728  4              IF .EVENT_ENTRY [EVENT$B_CMD_TYPE] EQL EVENT$K_TYPE_BREAK OR
;  3606        3729  4                 .EVENT_ENTRY [EVENT$B_CMD_TYPE] EQL EVENT$K_TYPE_TRACE
;  3607        3730  4              THEN
;  3608        3731  5                  BEGIN
;  3609        3732  5                  IF .PRIMARY[DBG$B_DHDR_TYPE] EQL DBG$K_PRIMARY_DESC
;  3610        3733  5                  THEN
;  3611        3734  6                      BEGIN
;  3612        3735  6                      IF .PRIMARY[DBG$B_DHDR_KIND] EQL RST$K_ROUTINE OR
;  3613        3736  6                         .PRIMARY[DBG$B_DHDR_KIND] EQL RST$K_ENTRY
;  3614        3737  6                      THEN
;  3615        3738  6                          EVENT_ENTRY [EVENT$V_BREAK_ADDRESS] = TRUE;
;  3616        3739  6
;  3617        3740  5                      END;
;  3618        3741  5
;  3619        3742  4                  END;
;  3620        3743  4
;  3621        3744  4              IF NOT DBG$NGET_ADDRESS(.PRIMARY, ADDRESS[0], ADR_TYP,
;  3622        3745  5                       (IF .EVENT_ENTRY [EVENT$V_BREAK_ADDRESS]
;  3623        3746  5                        THEN
;  3624        3747  5                            TRUE
;  3625        3748  5                        ELSE
;  3626        3749  4                            FALSE),
;  3627        3750  4                       .MESSAGE_VECT)
;  3628        3751  4              THEN
;  3629        3752  4                  RETURN STS$K_SEVERE;
;  3630        3753  4
;  3631        3754  4
;  3632        3755  4              EVENT_ENTRY [EVENT$L_ADDRESS] = .ADDRESS[0];
;  3633        3756  4              IF NOT .EVENT_ENTRY [EVENT$V_BREAK_ADDRESS]
;  3634        3757  4              THEN
;  3635        3758  5                  BEGIN
;  3636        3759  5                  IF DBG$IS_IT_ENTRY (.ADDRESS [0])
;  3637        3760  5                  THEN
;  3638        3761  5                      EVENT_ENTRY [EVENT$L_ADDRESS] = .ADDRESS[0] + 2;
;  3639        3762  5
;  3640        3763  4                  END;
;  3641        3764  4
;  3642        3765  4
;  3643        3766  4              ! Save the event's primary.
;  3644        3767  4              !
;  3645        3768  4              EVENT_ENTRY [EVENT$L_PRIMARY] = .NOUN_NODE [DBG$L_NOUN_VALUE];
;  3646        3769  4              DBG$NCOPY_DESC(.EVENT_ENTRY [EVENT$L_PRIMARY], PRIMARY, DUMMY, FALSE);
;  3647        3770  4
;  3648        3771  4
;  3649        3772  4              ! Now, get and save the VMS descriptor.
;  3650        3773  4              !
;  3651        3774  4              DBG$PRIM_TO_VAL(.PRIMARY, DBG$K_V_VALUE_DESC, VALUE);
;  3652        3775  4              CH$MOVE(T2, VALUE[DBG$A_VALUE_VMSDESC],
;  3653        3776  4                                  EVENT_ENTRY[EVENT$A_VMSDESC]);
;  3654        3777  4
;  3655        3778  4
;  3656        3779  4              ! Point to the next noun node.
;  3657        3780  4              !
;  3658        3781  4              NOUN_NODE = .NOUN_NODE[DBG$L_NOUN_LINK];
```

```
;  3659          3782  4
;  3660          3783  4
;  3661          3784  4                ! If there's more addresses, get an new event descriptor
;  3662          3785  4                ! from permanent memory, copy the current one to it, and
;  3663          3786  4                ! link it to the working queue; otherwise, exit this loop.
;  3664          3787  4                !
;  3665          3788  4                IF .NOUN_NODE NEQA 0
;  3666          3789  4                THEN
;  3667          3790  5                    BEGIN
;  3668          3791  5
;  3669          3792  5
;  3670          3793  5                    ! Get another event descriptor, and copy the old
;  3671          3794  5                    ! one into it.
;  3672          3795  5                    !
;  3673          3796  5                    EVENT_ENTRY = DBG$GET_MEMORY(EVENT$K_EVENT_DESCRIPTOR_SIZE);
;  3674          3797  5                    CH$MOVE(EVENT$K_EVENT_DESCRIPTOR_SIZE * 4,
;  3675          3798  5                            .EVENT_QUEUE [L_QUEUE_FLINK], .EVENT_ENTRY);
;  3676          3799  5
;  3677          3800  5
;  3678          3801  5                    ! Link the new one into the working queue.
;  3679          3802  5                    !
;  3680          3803  5                    INSQUE (.EVENT_ENTRY, EVENT_QUEUE);
;  3681          3804  5                    END
;  3682          3805  5
;  3683          3806  4                ELSE
;  3684          3807  4                    EXITLOOP;
;  3685          3808  4
;  3686          3809  4                END
;  3687          3810  4
;  3688          3811  2            END;
;  3689          3812  2
;  3690          3813  2
;  3691          3814  2            ! Now that we've built a working list of event entries, compare them to
;  3692          3815  2            ! the Command Queue, showing or deleting existing entries that match new
;  3693          3816  2            ! ones.
;  3694          3817  2            !
;  3695          3818  2            ! For each new event entry, check the existing entries in the
;  3696          3819  2            ! Command queue for any entries that match (command types, and
;  3697          3820  2            ! addresses if the command type is an access type).
;  3698          3821  2            !
;  3699          3822  2            ! So, while there are (still) entries on the working queue....
;  3700          3823  2            !
;  3701          3824  2            SHOWED_CANCELED_ANY = FALSE;
;  3702          3825  2            WHILE .EVENT_QUEUE [L_QUEUE_FLINK] NEQA EVENT_QUEUE DO
;  3703          3826  3                BEGIN
;  3704          3827  3
;  3705          3828  3
;  3706          3829  3                ! Point EVENT_ENTRY to the first new event entry on the
;  3707          3830  3                ! working queue.
;  3708          3831  3                !
;  3709          3832  3                EVENT_ENTRY = .EVENT_QUEUE [L_QUEUE_FLINK];
;  3710          3833  3
;  3711          3834  3
;  3712          3835  3                ! Point to the first and next existing entries in the Command Queue.
;  3713          3836  3                !
;  3714          3837  3                QUEUE_ENTRY = .EVENT$CMD_QUEUE [L_QUEUE_FLINK];
;  3715          3838  3
```

```
;  3716          3839  3
;  3717          3840  3          ! For each existing entry...
;  3718          3841  3          !
;  3719          3842  3          SHOWED_CANCELED_ONE = FALSE;
;  3720          3843  3          WHILE .QUEUE_ENTRY NEQA EVENT$CMD_QUEUE DO
;  3721          3844  4              BEGIN
;  3722          3845  4
;  3723          3846  4
;  3724          3847  4              ! Ignore 'deleted' entries that haven't been removed.
;  3725          3848  4              !
;  3726          3849  4              IF NOT .QUEUE_ENTRY [EVENT$V_DELETED]
;  3727          3850  4              THEN
;  3728          3851  5                  BEGIN
;  3729          3852  5
;  3730          3853  5
;  3731          3854  5                  ! Compare the command type with the existing one.
;  3732          3855  5                  !
;  3733          3856  5                  IF .EVENT_ENTRY [EVENT$B_CMD_TYPE] EQLU
;  3734          3857  5                     .QUEUE_ENTRY [EVENT$B_CMD_TYPE]
;  3735          3858  5                  THEN
;  3736          3859  6                      BEGIN
;  3737          3860  6
;  3738          3861  6
;  3739          3862  6                      ! We found an existing entry of the right type, so
;  3740          3863  6                      ! there is/are BREAK(s), TRACE(s), or WATCH(es) to
;  3741          3864  6                      ! be checked.
;  3742          3865  6                      !
;  3743          3866  6                      SHOWED_CANCELED_ANY = TRUE;
;  3744          3867  6
;  3745          3868  6
;  3746          3869  6                      ! If the /ALL qualifier was used, the show or delete
;  3747          3870  6                      ! this entry.  Otherwise, we've got more checking to
;  3748          3871  6                      ! do....
;  3749          3872  6                      !
;  3750          3873  6                      IF .SHOW_CANCEL_ALL
;  3751          3874  6                      THEN
;  3752          3875  7                          SHOW_OR_DELETE_EVENT_ENTRY (QUEUE_ENTRY)
;  3753          3876  7
;  3754          3877  6                      ELSE
;  3755          3878  7                          BEGIN
;  3756          3879  7
;  3757          3880  7
;  3758          3881  7                          ! Compare the command kind of this existing entry
;  3759          3882  7                          ! and the new one.
;  3760          3883  7                          !
;  3761          3884  7                          IF .EVENT_ENTRY [EVENT$B_CMD_KIND] EQLU
;  3762          3885  7                             .QUEUE_ENTRY [EVENT$B_CMD_KIND]
;  3763          3886  7                          THEN
;  3764          3887  8                              BEGIN
;  3765          3888  8
;  3766          3889  8
;  3767          3890  8                              ! The command kinds match.  Select on the kind.
;  3768          3891  8                              !
;  3769          3892  8                              SELECTONE .EVENT_ENTRY [EVENT$B_CMD_KIND] OF
;  3770          3893  8                                  SET
;  3771          3894  8
;  3772          3895  8
```

```
;  3773          3896  8                    ! The command kind is access (/READ, /WRITE,
;  3774          3897  8                    ! /MODIFY, /EXECUTE, or /RETURN).  We now have
;  3775          3898  8                    ! to compare the addresses of the entries to
;  3776          3899  8                    ! determine if this existing one must be deleted.
;  3777          3900  8                    !
;  3778          3901  8                    [EVENT$K_KIND_ACC]:
;  3779          3902  9                        BEGIN
;  3780          3903  9                        LOCAL CODE_1,CODE_2;
;  3781          3904  9
;  3782          3905  9                        ! If either of the entries is /EXECUTE or
;  3783          3906  9                        ! /RETURN, we'll compare the byte addresses,
;  3784          3907  9                        ! otherwise compare the two VMS descriptors,
;  3785          3908  9                        ! and delete the exisiting entry if they're
;  3786          3909  9                        ! the same.
;  3787          3910  9                        !
;  3788          3911  9                        CODE_1 = .EVENT_ENTRY [EVENT$B_SUB_KIND];
;  3789          3912  9                        CODE_2 = .QUEUE_ENTRY [EVENT$B_SUB_KIND];
;  3790          3913 11                        IF (IF (.CODE_1 EQLU EVENT$K_ACC_EXEC)
;  3791          3914 11                            OR (.CODE_1 EQLU EVENT$K_ACC_RTRN)
;  3792          3915 11                            OR (.CODE_2 EQLU EVENT$K_ACC_EXEC)
;  3793          3916 11                            OR (.CODE_2 EQLU EVENT$K_ACC_RTRN)
;  3794          3917 10                            THEN
;  3795          3918 10                                .EVENT_ENTRY [EVENT$L_ADDRESS] EQLA
;  3796          3919 10                                .QUEUE_ENTRY [EVENT$L_ADDRESS]
;  3797          3920 10                            ELSE
;  3798          3921 10                                COMPARE_VMSDESC(EVENT_ENTRY [EVENT$A_VMSDESC],
;  3799          3922 10                                                QUEUE_ENTRY [EVENT$A_VMSDESC])
;  3800          3923 10                        )
;  3801          3924  9                        THEN
;  3802          3925  9                            SHOW_OR_DELETE_EVENT_ENTRY (QUEUE_ENTRY);
;  3803          3926  8                        END;
;  3804          3927  8
;  3805          3928  8                    ! The command kind is instruction.  Compare the two
;  3806          3929  8                    ! entries' subkind, and delete the existing entry if
;  3807          3930  8                    ! they're the same.
;  3808          3931  8                    !
;  3809          3932  8                    [EVENT$K_KIND_INS]:
;  3810          3933  8                        CASE .EVENT_ENTRY [EVENT$B_SUB_KIND]
;  3811          3934  8                        FROM EVENT$K_INS_CALL TO EVENT$K_INS_USER OF
;  3812          3935  8                            SET
;  3813          3936  8                            [EVENT$K_INS_CALL TO EVENT$K_INS_LINE]:
;  3814          3937  8                                BEGIN
;  3815          3938  9                                IF .EVENT_ENTRY [EVENT$B_SUB_KIND] EQLU
;  3816          3939  9                                    .QUEUE_ENTRY [EVENT$B_SUB_KIND]
;  3817          3940  9                                THEN
;  3818          3941  9                                    SHOW_OR_DELETE_EVENT_ENTRY (QUEUE_ENTRY);
;  3819          3942  9                                END;
;  3820          3943  8                            [EVENT$K_INS_EVRY TO EVENT$K_INS_USER]:
;  3821          3944  8                                BEGIN
;  3822          3945  9                                IF (.QUEUE_ENTRY [EVENT$B_SUB_KIND] EQLU
;  3823          3946 10                                    EVENT$K_INS_EVRY) OR
;  3824          3947  9                                   (.QUEUE_ENTRY [EVENT$B_SUB_KIND] EQLU
;  3825          3948 10                                    EVENT$K_INS_USER)
;  3826          3949 10                                THEN
;  3827          3950  9                                    SHOW_OR_DELETE_EVENT_ENTRY (QUEUE_ENTRY);
;  3828          3951  9                                END;
;  3829          3952  8
```

```
;  3830     3953  8                                TES;
;  3831     3954  8
;  3832     3955  8
;  3833     3956  8                            ! The access type is exception, so delete the existing
;  3834     3957  8                            ! entry.
;  3835     3958  8                            !
;  3836     3959  8                            [EVENT$K_KIND_EXC]:
;  3837     3960  8                                SHOW_OR_DELETE_EVENT_ENTRY (QUEUE_ENTRY);
;  3838     3961  8
;  3839     3962  8                            ! The command kind is invalid - yell !
;  3840     3963  8                            !
;  3841     3964  8                            [OTHERWISE]:
;  3842     3965  8                                $DBG_ERROR('DBGEVENT\EVENT_SHOW_CANCEL_SEMANTICS 40');
;  3843     3966  8
;  3844     3967  8                            TES;
;  3845     3968  8
;  3846     3969  8                        END;
;  3847     3970  7
;  3848     3971  7                    END;
;  3849     3972  6
;  3850     3973  6                END;
;  3851     3974  5
;  3852     3975  5            END;
;  3853     3976  4
;  3854     3977  4
;  3855     3978  4            ! Point to the next Command Queue entries.
;  3856     3979  4            !
;  3857     3980  4            QUEUE_ENTRY = .QUEUE_ENTRY [EVENT$L_CMD_FLINK];
;  3858     3981  4            END;
;  3859     3982  3
;  3860     3983  3
;  3861     3984  3            ! If this entry wasn't shown or cancelled, announce the fact, but we
;  3862     3985  3            ! did find one of this type, say we couldn't find this one.
;  3863     3986  3            !
;  3864     3987  3            IF NOT .SHOWED_CANCELED_ONE AND .SHOWED_CANCELED_ANY
;  3865     3988  3            THEN
;  3866     3989  3                SIGNAL      (SELECTONE .VERB_NODE [DBG$B_VERB_COMPOSITE] OF
;  3867     3990  3                                SET
;  3868     3991  3                                [EVENT$K_SHOW_BREAK, EVENT$K_CANCEL_BREAK,
;  3869     3992  3                                 EVENT$K_CANCEL_BREAK_EXC]:
;  3870     3993  3                                    DBG$_NOSUCHBPT;
;  3871     3994  3                                [EVENT$K_SHOW_TRACE, EVENT$K_CANCEL_TRACE]:
;  3872     3995  3                                    DBG$_NOSUCHTPT;
;  3873     3996  3                                [EVENT$K_SHOW_WATCH, EVENT$K_CANCEL_WATCH]:
;  3874     3997  3                                    DBG$_NOSUCHWPT;
;  3875     3998  3                                TES
;  3876     3999  3                            );
;  3877     4000  3
;  3878     4001  3
;  3879     4002  3            ! Remove this new entry from the working queue.
;  3880     4003  3            !
;  3881     4004  3            REMQUE (.EVENT_ENTRY, EVENT_ENTRY);
;  3882     4005  3            DBG$REL_MEMORY (.EVENT_ENTRY);
;  3883     4006  3            END;
;  3884     4007  2
;  3885     4008  2
;  3886     4009  2
```

```
;  3887          4010  2     ! If we couldn't find an entry of the right type, announce the fact.
;  3888          4011  2     !
;  3889          4012  2     IF NOT .SHOWED_CANCELED_ANY
;  3890          4013  2     THEN
;  3891          4014  2         SIGNAL   (SELECTONE .VERB_NODE [DBG$B_VERB_COMPOSITE] OF
;  3892          4015  2                       SET
;  3893          4016  2                       [EVENT$K_SHOW_BREAK, EVENT$K_CANCEL_BREAK,
;  3894          4017  2                        EVENT$K_CANCEL_BREAK_EXC]:
;  3895          4018  2                          DBG$_NOBREAKS;
;  3896          4019  2                       [EVENT$K_SHOW_TRACE, EVENT$K_CANCEL_TRACE]:
;  3897          4020  2                          DBG$_NOTRACES;
;  3898          4021  2                       [EVENT$K_SHOW_WATCH, EVENT$K_CANCEL_WATCH]:
;  3899          4022  2                          DBG$_NOWATCHES;
;  3900          4023  2                       TES
;  3901          4024  2                      );
;  3902          4025  2
;  3903          4026  2
;  3904          4027  2     ! We're done, bye.
;  3905          4028  2     !
;  3906          4029  2     RETURN STS$K_SUCCESS;
;  3907          4030  1     END;


                                                                    .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

54  4E  45  56  45  5C  54  4E  45  56  45  47  42  44  27  001D9 P.ACD:  .ASCII  \'DBGEVENT\<92>\EVENT_SHOW_CANCEL_SEMANT\  ;
45  53  5F  4C  45  43  4E  41  43  5F  57  4F  48  53  5F  001E8                                                            ;
                                    54  4E  41  4D  001F7                                                               ;
                            30  31  20  53  43  49  001FB          .ASCII  \ICS 10\                                      ;
54  4E  45  56  45  5C  54  4E  45  56  45  47  42  44  27  00201 P.ACE:  .ASCII  \'DBGEVENT\<92>\EVENT_SHOW_CANCEL_SEMANT\  ;
45  53  5F  4C  45  43  4E  41  43  5F  57  4F  48  53  5F  00210                                                            ;
                                    54  4E  41  4D  0021F                                                               ;
                            30  32  20  53  43  49  00223          .ASCII  \ICS 20\                                      ;
54  4E  45  56  45  5C  54  4E  45  56  45  47  42  44  27  00229 P.ACF:  .ASCII  \'DBGEVENT\<92>\EVENT_SHOW_CANCEL_SEMANT\  ;
45  53  5F  4C  45  43  4E  41  43  5F  57  4F  48  53  5F  00238                                                            ;
                                    54  4E  41  4D  00247                                                               ;
                            30  33  20  53  43  49  0024B          .ASCII  \ICS 30\                                      ;
54  4E  45  56  45  5C  54  4E  45  56  45  47  42  44  27  00251 P.ACG:  .ASCII  \'DBGEVENT\<92>\EVENT_SHOW_CANCEL_SEMANT\  ;
45  53  5F  4C  45  43  4E  41  43  5F  57  4F  48  53  5F  00260                                                            ;
                                    54  4E  41  4D  0026F                                                               ;
                            30  34  20  53  43  49  00273          .ASCII  \ICS 40\                                      ;


                                                                    .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                                OFFC 00000          .ENTRY  DBG$EVENT_SHOW_CANCEL_SEMANTICS, Save R2,-  ; 3497
                                                                            R3,R4,R5,R6,R7,R8,R9,R10,R11
                        5B 00000000'    EF  9E 00002            MOVAB   EVENT$CMD_QUEUE, R11                            ;
                        5A 00000000'    EF  9E 00009            MOVAB   P.ACD, R10                                      ;
                        59 00000000G    00  9E 00010            MOVAB   LIB$SIGNAL, R9                                  ;
                        5E              28  C2 00017            SUBL2   #40, SP                                         ;
                    18  AE          18  AE  9E 0001A            MOVAB   EVENT_QUEUE, EVENT_QUEUE                        ; 3562
                    1C  AE          18  AE  9E 0001F            MOVAB   EVENT_QUEUE, EVENT_QUEUE+4                      ; 3563
                                        10  DD 00024            PUSHL   #16                                             ; 3569
                00000000G   00          01  FB 00026            CALLS   #1, DBG$GET_MEMORY                              ;
```

```
                56              50  D0 0002D            MOVL    R0, EVENT_ENTRY
              18 AE             66  0E 00030            INSQUE  (EVENT_ENTRY), EVENT_QUEUE                  ; 3570
                53          14  A6  9E 00034            MOVAB   20(EVENT_ENTRY), R3                         ; 3575
                            01  A3  94 00038            CLRB    1(R3)
                52          04  AC  D0 0003B            MOVL    VERB_NODE, R2                               ; 3580
                50          01  A2  9A 0003F            MOVZBL  1(R2), R0
                                09  15 00043            BLEQ    1$                                          ; 3583
                            02      50  91 00045        CMPB    R0, #2
                                    04  1A 00048        BGTRU   1$
                                    63  94 0004A        CLRB    (R3)                                        ; 3584
                                    32  11 0004C        BRB     6$
                            04      50  91 0004E 1$:    CMPB    R0, #4                                      ; 3586
                                    07  12 00051        BNEQ    2$
                            63  0200    8F  B0 00053    MOVW    #512, (R3)                                  ; 3588
                                    26  11 00058        BRB     6$                                          ; 3580
                            09      50  91 0005A 2$:    CMPB    R0, #9                                      ; 3592
                                    05  13 0005D        BEQL    3$
                            0B      50  91 0005F        CMPB    R0, #11
                                    05  12 00062        BNEQ    4$
                            63      01  90 00064 3$:    MOVB    #1, (R3)                                    ; 3593
                                    17  11 00067        BRB     6$
                            0E      50  91 00069 4$:    CMPB    R0, #14                                     ; 3595
                                    05  12 0006C        BNEQ    5$
                            63      02  90 0006E        MOVB    #2, (R3)                                    ; 3596
                                    0D  11 00071        BRB     6$
                                    5A  DD 00073 5$:    PUSHL   R10                                         ; 3599
                                    01  DD 00075        PUSHL   #1
                        00028362    8F  DD 00077        PUSHL   #164706
                            69      03  FB 0007D        CALLS   #3, LIB$SIGNAL
              03 A3                 08  8A 00080 6$:    BICB2   #8, 3(R3)                                   ; 3606
                        28  A6      D4  00084            CLRL    40(EVENT_ENTRY)                             ; 3611
                            57  08  A2  D0 00087        MOVL    8(R2), NOUN_NODE                            ; 3616
                            50  04  A2  D0 0008B        MOVL    4(R2), ADVERB_NODE                          ; 3621
                                    58  D4 0008F        CLRL    SHOW_CANCEL_ALL                             ; 3626
                                    50  D5 00091        TSTL    ADVERB_NODE                                 ; 3631
                                    7C  13 00093        BEQL    17$
                            0B      60  91 00095        CMPB    (ADVERB_NODE), #11                          ; 3645
                                    06  12 00098        BNEQ    7$
              01 A3                 02  90 0009A        MOVB    #2, 1(R3)                                   ; 3646
                                    71  11 0009E        BRB     17$
                            0A      60  91 000A0 7$:    CMPB    (ADVERB_NODE), #10                          ; 3652
                                    0A  13 000A3        BEQL    8$
                            12      60  91 000A5        CMPB    (ADVERB_NODE), #18
                                    4F  1F 000A8        BLSSU   15$
                            16      60  91 000AA        CMPB    (ADVERB_NODE), #22
                                    4A  1A 000AD        BGTRU   15$
              01 A3                 01  90 000AF 8$:    MOVB    #1, 1(R3)                                   ; 3659
                            12      60  91 000B3        CMPB    (ADVERB_NODE), #18                          ; 3667
                                    05  12 000B6        BNEQ    9$
                            50      05  D0 000B8        MOVL    #5, R0
                                    36  11 000BB        BRB     14$
                            13      60  91 000BD 9$:    CMPB    (ADVERB_NODE), #19                          ; 3669
                                    05  12 000C0        BNEQ    10$
                            50      06  D0 000C2        MOVL    #6, R0
                                    2C  11 000C5        BRB     14$
                            14      60  91 000C7 10$:   CMPB    (ADVERB_NODE), #20                          ; 3671
                                    05  12 000CA        BNEQ    11$
```

```
                50              07  D0 000CC          MOVL    #7, R0                                  ;
                                22  11 000CF          BRB     14$                                     ;
                16              60  91 000D1 11$:     CMPB    (ADVERB_NODE), #22                      ; 3673
                                05  12 000D4          BNEQ    12$                                     ;
                50              09  D0 000D6          MOVL    #9, R0                                  ;
                                18  11 000D9          BRB     14$                                     ;
                15              60  91 000DB 12$:     CMPB    (ADVERB_NODE), #21                      ; 3675
                                05  12 000DE          BNEQ    13$                                     ;
                50              08  D0 000E0          MOVL    #8, R0                                  ;
                                0E  11 000E3          BRB     14$                                     ;
                        28      AA  9F 000E5 13$:     PUSHAB  P.ACE                                   ; 3678
                                01  DD 000E8          PUSHL   #1                                      ;
                00028362        8F  DD 000EA          PUSHL   #164706                                 ;
                69              03  FB 000F0          CALLS   #3, LIB$SIGNAL                          ;
            02  A3              50  90 000F3 14$:     MOVB    R0, 2(R3)                               ; 3665
                                18  11 000F7          BRB     17$                                     ; 3664
                17              60  91 000F9 15$:     CMPB    (ADVERB_NODE), #23                      ; 3686
                                05  12 000FC          BNEQ    16$                                     ;
                58              01  D0 000FE          MOVL    #1, SHOW_CANCEL_ALL                     ; 3687
                                0E  11 00101          BRB     17$                                     ;
                        50      AA  9F 00103 16$:     PUSHAB  P.ACF                                   ; 3693
                                01  DD 00106          PUSHL   #1                                      ;
                00028362        8F  DD 00108          PUSHL   #164706                                 ;
                69              03  FB 0010E          CALLS   #3, LIB$SIGNAL                          ;
                                57  D5 00111 17$:     TSTL    NOUN_NODE                               ; 3704
                                0B  12 00113          BNEQ    19$                                     ;
                0B              62  91 00115          CMPB    (R2), #11                               ; 3707
                                03  12 00118          BNEQ    18$                                     ;
                58              01  D0 0011A          MOVL    #1, SHOW_CANCEL_ALL                     ; 3709
                            0DC1  31  11 0011D 18$:     BRW     27$                                     ; 3704
            08  AE                  67  D0 00120 19$:     MOVL    (NOUN_NODE), PRIMARY                    ; 3726
            19  A6                  20  8A 00124          BICB2   #32, 25(EVENT_ENTRY)                    ; 3727
                        14      A6  95  00128          TSTB    20(EVENT_ENTRY)                         ; 3728
                                    06  13 0012B          BEQL    20$                                     ;
                01      14      A6  91  0012D          CMPB    20(EVENT_ENTRY), #1                     ; 3729
                                    1B  12 00131          BNEQ    22$                                     ;
                50      08      AE  D0  00133 20$:     MOVL    PRIMARY, R0                             ; 3732
            79  8F      02      A0  91  00137          CMPB    2(R0), #121                             ;
                                    10  12 0013C          BNEQ    22$                                     ;
                02      07      A0  91  0013E          CMPB    7(R0), #2                               ; 3735
                                    06  13 00142          BEQL    21$                                     ;
                08      07      A0  91  00144          CMPB    7(R0), #8                               ; 3736
                                    04  12 00148          BNEQ    22$                                     ;
            19  A6                  20  88 0014A 21$:     BISB2   #32, 25(EVENT_ENTRY)                    ; 3738
                        08      AC  DD  0014E 22$:     PUSHL   MESSAGE_VECT                            ; 3750
        04  19  A6                  05  E1 00151          BBC     #5, 25(EVENT_ENTRY), 23$                ; 3745
                                    01  DD 00156          PUSHL   #1                                      ;
                                    02  11 00158          BRB     24$                                     ;
                                    7E  D4 0015A 23$:     CLRL    -(SP)                                   ;
                        08      AE  9F  0015C 24$:     PUSHAB  ADR_TYP                                 ; 3744
                        1C      AE  9F  0015F          PUSHAB  ADDRESS                                 ;
                        18      AE  DD  00162          PUSHL   PRIMARY                                 ;
        00000000G       00          05  FB 00165          CALLS   #5, DBG$NGET_ADDRESS                    ;
                        04          50  E8 0016C          BLBS    R0, 25$                                 ;
                        50          04  D0 0016F          MOVL    #4, R0                                  ; 3752
                                        04  00172          RET                                             ;
            2C  A6      10      AE  D0  00173 25$:     MOVL    ADDRESS, 44(EVENT_ENTRY)                ; 3755
```

```
                13      19  A6              05  E0 00178          BBS     #5, 25(EVENT_ENTRY), 26$                ; 3756
                                        10  AE  DD 0017D          PUSHL   ADDRESS                                 ; 3759
                    00000000G   00          01  FB 00180          CALLS   #1, DBG$IS_IT_ENTRY                     ;
                                06          50  E9 00187          BLBC    R0, 26$                                 ;
            2C  A6      10  AE              02  C1 0018A          ADDL3   #2, ADDRESS, 44(EVENT_ENTRY)            ; 3761
                        28  A6              67  D0 00190 26$:     MOVL    (NOUN_NODE), 40(EVENT_ENTRY)            ; 3768
                                            7E  D4 00194          CLRL    -(SP)                                   ; 3769
                                        08  AE  9F 00196          PUSHAB  DUMMY                                   ;
                                        10  AE  9F 00199          PUSHAB  PRIMARY                                 ;
                                        28  A6  DD 0019C          PUSHL   40(EVENT_ENTRY)                         ;
                    00000000G   00              04  FB 0019F      CALLS   #4, DBG$NCOPY_DESC                      ;
                                        0C  AE  9F 001A6          PUSHAB  VALUE                                   ; 3774
                                7E              83  8F 9A 001A9  MOVZBL  #131, -(SP)                            ;
                                        10  AE  DD 001AD          PUSHL   PRIMARY                                 ;
                    00000000G   00              03  FB 001B0      CALLS   #3, DBG$PRIM_TO_VAL                     ;
                                50          0C  AE  D0 001B7      MOVL    VALUE, R0                               ; 3775
            34  A6      14  A0              0C  28 001BB          MOVC3   #12, 20(R0), 52(EVENT_ENTRY)            ; 3776
                                57      08  A7  D0 001C1          MOVL    8(NOUN_NODE), NOUN_NODE                 ; 3781
                                            1A  13 001C5          BEQL    27$                                     ; 3788
                                            10  DD 001C7          PUSHL   #16                                     ; 3796
                    00000000G   00              01  FB 001C9      CALLS   #1, DBG$GET_MEMORY                      ;
                                56              50  D0 001D0      MOVL    R0, EVENT_ENTRY                         ;
                66      18  BE          0040    8F  28 001D3      MOVC3   #64, @EVENT_QUEUE, (EVENT_ENTRY)        ; 3798
                        18  AE              66  0E 001DA          INSQUE  (EVENT_ENTRY), EVENT_QUEUE              ; 3803
                                        FF3F    31  001DE         BRW     19$                                     ; 3788
                                            55  D4 001E1 27$:     CLRL    SHOWED_CANCELED_ANY                     ; 3824
                                50      18  AE  9E 001E3 28$:     MOVAB   EVENT_QUEUE, R0                         ; 3825
                                50      18  AE  D1 001E7          CMPL    EVENT_QUEUE, R0                         ;
                                            03  12 001EB          BNEQ    29$                                     ;
                                        011B    31  001ED         BRW     56$                                     ;
                                56      18  AE  D0 001F0 29$:     MOVL    EVENT_QUEUE, EVENT_ENTRY                ; 3832
                                52          6B  D0 001F4          MOVL    EVENT$CMD_QUEUE, QUEUE_ENTRY            ; 3837
                                            54  D4 001F7          CLRL    SHOWED_CANCELED_ONE                     ; 3842
                                50          6B  9E 001F9 30$:     MOVAB   EVENT$CMD_QUEUE, R0                     ; 3843
                                50          52  D1 001FC          CMPL    QUEUE_ENTRY, R0                         ;
                                            03  12 001FF          BNEQ    31$                                     ;
                                        00B3    31  00201         BRW     47$                                     ;
                                53      14  A2  9E 00204 31$:     MOVAB   20(QUEUE_ENTRY), R3                     ; 3849
                                            63  D5 00208          TSTL    (R3)                                    ;
                                            03  18 0020A          BGEQ    33$                                     ;
                                        00A2    31  0020C 32$:     BRW     46$                                     ;
                                63      14  A6  91 0020F 33$:     CMPB    20(EVENT_ENTRY), (R3)                   ; 3857
                                            F7  12 00213          BNEQ    32$                                     ;
                                55              01  D0 00215      MOVL    #1, SHOWED_CANCELED_ANY                 ; 3866
                                6F              58  E8 00218      BLBS    SHOW_CANCEL_ALL, 42$                    ; 3873
                                50      15  A6  9A 0021B          MOVZBL  21(EVENT_ENTRY), R0                     ; 3884
                                50      01  A3  91 0021F          CMPB    1(R3), R0                               ; 3885
                                            E7  12 00223          BNEQ    32$                                     ;
                                            50  D5 00225          TSTL    R0                                      ; 3901
                                            35  12 00227          BNEQ    37$                                     ;
                                50      16  A6  9A 00229          MOVZBL  22(EVENT_ENTRY), CODE_1                 ; 3911
                                51      02  A3  9A 0022D          MOVZBL  2(R3), CODE_2                           ; 3912
                                03          50  D1 00231          CMPL    CODE_1, #3                              ; 3913
                                            0F  13 00234          BEQL    34$                                     ;
                                04          50  D1 00236          CMPL    CODE_1, #4                              ; 3914
                                            0A  13 00239          BEQL    34$                                     ;
                                03          51  D1 0023B          CMPL    CODE_2, #3                              ; 3915
```

```
                                   05  13 0023E            BEQL    34$                                          ;
                        04         51  D1 00240            CMPL    CODE_2, #4                                   ; 3916
                                   09  12 00243            BNEQ    36$                                          ;
               2C       A2     2C  A6  D1 00245  34$:      CMPL    44(EVENT_ENTRY), 44(QUEUE_ENTRY)             ; 3919
                                   65  12 0024A  35$:      BNEQ    46$                                          ;
                                   3C  11 0024C            BRB     42$                                          ;
                               34  A2  9F 0024E  36$:      PUSHAB  52(QUEUE_ENTRY)                              ; 3922
                               34  A6  9F 00251            PUSHAB  52(EVENT_ENTRY)                              ; 3921
            0000V       CF         02  FB 00254            CALLS   #2, COMPARE_VMSDESC                          ; 3922
                        55         50  E9 00259            BLBC    R0, 46$                                      ;
                                   2C  11 0025C            BRB     42$                                          ; 3925
                        01         50  91 0025E  37$:      CMPB    R0, #1                                       ; 3933
                                   22  12 00261            BNEQ    41$                                          ;
                 04     05     16  A6  8F 00263            CASEB   22(EVENT_ENTRY), #5, #4                      ; 3934
 0011          000A         000A     000A 00268  38$:      .WORD   39$-38$,-                                    ;
                                     0011 00270                    39$-38$,-                                    ;
                                                                   39$-38$,-                                    ;
                                                                   40$-38$,-                                    ;
                                                                   40$-38$                                      ;
               02       A3     16  A6  91 00272  39$:      CMPB    22(EVENT_ENTRY), 2(R3)                       ; 3940
                                   D1  11 00277            BRB     35$                                          ;
                        08     02  A3  91 00279  40$:      CMPB    2(R3), #8                                    ; 3946
                                   0B  13 0027D            BEQL    42$                                          ;
                        09     02  A3  91 0027F            CMPB    2(R3), #9                                    ; 3948
                                   C5  11 00283            BRB     35$                                          ;
                        02         50  91 00285  41$:      CMPB    R0, #2                                       ; 3959
                                   19  12 00288            BNEQ    45$                                          ;
                        0B     04  BC  91 0028A  42$:      CMPB    @VERB_NODE, #11                              ; 3960
                                   09  12 0028E            BNEQ    43$                                          ;
                                   52  DD 00290            PUSHL   QUEUE_ENTRY                                  ;
            0000V       CF         01  FB 00292            CALLS   #1, SHOW_EVENT_ENTRY                         ;
                                   05  11 00297            BRB     44$                                          ;
               03       A3     80  8F  88 00299  43$:      BISB2   #128, 3(R3)                                  ;
                        54         01  D0 0029E  44$:      MOVL    #1, SHOWED_CANCELED_ONE                      ;
                                   0E  11 002A1            BRB     46$                                          ; 3892
                               78  AA  9F 002A3  45$:      PUSHAB  P.ACG                                        ; 3966
                                   01  DD 002A6            PUSHL   #1                                           ;
                     00028362      8F  DD 002A8            PUSHL   #164706                                      ;
                        69         03  FB 002AE            CALLS   #3, LIB$SIGNAL                               ;
                        52         62  D0 002B1  46$:      MOVL    (QUEUE_ENTRY), QUEUE_ENTRY                   ; 3981
                                 FF42  31 002B4            BRW     30$                                          ; 3843
                        42         54  E8 002B7  47$:      BLBS    SHOWED_CANCELED_ONE, 55$                     ; 3988
                        3F         55  E9 002BA            BLBC    SHOWED_CANCELED_ANY, 55$                     ;
   50          04   BC  08         08  EF 002BD            EXTZV   #8, #8, @VERB_NODE, R0                       ; 3990
                                   05  15 002C3            BLEQ    48$                                          ; 3992
                        02         50  91 002C5            CMPB    R0, #2                                       ;
                                   05  1B 002C8            BLEQU   49$                                          ;
                        04         50  91 002CA  48$:      CMPB    R0, #4                                       ;
                                   08  12 002CD            BNEQ    50$                                          ;
                     0002801B      8F  DD 002CF  49$:      PUSHL   #163867                                      ;
                                   22  11 002D5            BRB     54$                                          ;
                        09         50  91 002D7  50$:      CMPB    R0, #9                                       ; 3995
                                   05  13 002DA            BEQL    51$                                          ;
                        0B         50  91 002DC            CMPB    R0, #11                                      ;
                                   08  12 002DF            BNEQ    52$                                          ;
                     00028023      8F  DD 002E1  51$:      PUSHL   #163875                                      ;
                                   10  11 002E7            BRB     54$                                          ;
```

```
                                 0E              50  91 002E9 52$:    CMPB    R0, #14                                 ;3997
                                                 05  13 002EC          BEQL    53$
                                 7E              01  CE 002EE          MNEGL   #1, -(SP)
                                                 06  11 002F1          BRB     54$
                                     0002802B    8F  DD 002F3 53$:    PUSHL   #163883
                                 69              01  FB 002F9 54$:    CALLS   #1, LIB$SIGNAL                          ;3990
                                 56              66  0F 002FC 55$:    REMQUE  (EVENT_ENTRY), EVENT_ENTRY              ;4005
                                                 56  DD 002FF          PUSHL   EVENT_ENTRY                             ;4006
                          00000000G  00          01  FB 00301          CALLS   #1, DBG$REL_MEMORY
                                                 FED8 31 00308         BRW     28$                                     ;3825
                                 3F              55  E8 0030B 56$:    BLBS    SHOWED_CANCELED_ANY, 64$                ;4012
        50      04      BC       08              08  EF 0030E          EXTZV   #8, #8, @VERB_NODE, R0                  ;4014
                                                 05  15 00314          BLEQ    57$                                     ;4016
                                 02              50  91 00316          CMPB    R0, #2
                                                 05  18 00319          BLEQU   58$
                                 04              50  91 0031B 57$:    CMPB    R0, #4
                                                 08  12 0031E          BNEQ    59$
                                     00028013    8F  DD 00320 58$:    PUSHL   #163859
                                                 22  11 00326          BRB     63$
                                 09              50  91 00328 59$:    CMPB    R0, #9                                  ;4019
                                                 05  13 0032B          BEQL    60$
                                 0B              50  91 0032D          CMPB    R0, #11
                                                 08  12 00330          BNEQ    61$
                                     00028033    8F  DD 00332 60$:    PUSHL   #163891
                                                 10  11 00338          BRB     63$
                                 0E              50  91 0033A 61$:    CMPB    R0, #14                                 ;4021
                                                 05  13 0033D          BEQL    62$
                                 7E              01  CE 0033F          MNEGL   #1, -(SP)
                                                 06  11 00342          BRB     63$
                                     0002803B    8F  DD 00344 62$:    PUSHL   #163899
                             69                  01  FB 0034A 63$:    CALLS   #1, LIB$SIGNAL                          ;4014
                             50                  01  D0 0034D 64$:    MOVL    #1, R0                                  ;4029
                                                     04 00350          RET                                             ;4030

; Routine Size:  849 bytes,    Routine Base:  DBG$CODE + 15E8
```

```
;  3909     4031  1 GLOBAL ROUTINE DBG$EVENT_CANCEL_ALL: NOVALUE =
;  3910     4032  1 !
;  3911     4033  1 ! FUNCTION
;  3912     4034  1 !       This routine cancels all eventpoints.
;  3913     4035  1 !
;  3914     4036  1 ! INPUTS
;  3915     4037  1 !       NONE
;  3916     4038  1 !
;  3917     4039  1 ! OUTPUTS
;  3918     4040  1 !       All command event entries are deleted from the command queue.
;  3919     4041  1 !
;  3920     4042  1
;  3921     4043  1     BEGIN
;  3922     4044  1     LOCAL
;  3923     4045  1         EVENT_ENTRY :   REF EVENT$EVENT_DESCRIPTOR; ! Event entry pointer
;  3924     4046  1
;  3925     4047  1
;  3926     4048  1
;  3927     4049  1     ! Starting at the head of the command queue, 'delete' the command
;  3928     4050  1     ! entries.
;  3929     4051  1     !
;  3930     4052  1     EVENT_ENTRY = .EVENT$CMD_QUEUE [L_QUEUE_FLINK];
;  3931     4053  1     WHILE .EVENT_ENTRY NEQA EVENT$CMD_QUEUE DO
;  3932     4054  1         BEGIN
;  3933     4055  1
;  3934     4056  1
;  3935     4057  1         ! Ignore 'deleted' entries that haven't been removed.
;  3936     4058  1         !
;  3937     4059  1         IF NOT .EVENT_ENTRY [EVENT$V_DELETED]
;  3938     4060  1         THEN
;  3939     4061  1
;  3940     4062  1
;  3941     4063  1             ! If this is a BREAK, TRACE, or WATCH command, delete it.
;  3942     4064  1             !
;  3943     4065  1             IF (.EVENT_ENTRY [EVENT$B_CMD_TYPE] EQLU
;  3944     4066  1                 EVENT$K_TYPE_BREAK
;  3945     4067  1                ) OR
;  3946     4068  1                (.EVENT_ENTRY [EVENT$B_CMD_TYPE] EQLU
;  3947     4069  1                 EVENT$K_TYPE_TRACE
;  3948     4070  1                ) OR
;  3949     4071  1                (.EVENT_ENTRY [EVENT$B_CMD_TYPE] EQLU
;  3950     4072  1                 EVENT$K_TYPE_WATCH
;  3951     4073  1                )
;  3952     4074  1             THEN
;  3953     4075  1                 EVENT_ENTRY [EVENT$V_DELETED] = TRUE;
;  3954     4076  1
;  3955     4077  1
;  3956     4078  1         ! Point to the next Command Queue entries.
;  3957     4079  1         !
;  3958     4080  1         EVENT_ENTRY = .EVENT_ENTRY [EVENT$L_CMD_FLINK];
;  3959     4081  1         END;
;  3960     4082  1     END;
```

```
                                 0004 00000          .ENTRY  DBG$EVENT_CANCEL_ALL, Save R2                   ; 4031
                     52 00000000'  EF   9E 00002     MOVAB   EVENT$CMD_QUEUE, R2                            ;
                     50            62   D0 00009     MOVL    EVENT$CMD_QUEUE, EVENT_ENTRY                   ; 4052
                     51            62   9E 0000C 1$: MOVAB   EVENT$CMD_QUEUE, R1                           ; 4053
                     51            50   D1 0000F     CMPL    EVENT_ENTRY, R1                                ;
                                   1F   13 00012     BEQL    4$                                             ;
                           17      A0   95 00014     TSTB    23(EVENT_ENTRY)                                ; 4059
                                   15   19 00017     BLSS    3$                                             ;
                     51    14      A0   9A 00019     MOVZBL  20(EVENT_ENTRY), R1                            ; 4065
                                   0A   13 0001D     BEQL    2$                                             ;
                     01            51   91 0001F     CMPB    R1, #1                                         ; 4068
                                   05   13 00022     BEQL    2$                                             ;
                     02            51   91 00024     CMPB    R1, #2                                         ; 4071
                                   05   12 00027     BNEQ    3$                                             ;
               17    A0      80    8F   88 00029 2$: BISB2   #128, 23(EVENT_ENTRY)                         ; 4075
                     50            60   D0 0002E 3$: MOVL    (EVENT_ENTRY), EVENT_ENTRY                     ; 4080
                                   D9   11 00031     BRB     1$                                             ; 4053
                                        04 00033 4$: RET                                                    ; 4082

; Routine Size:  52 bytes,    Routine Base:  DBG$CODE + 1939
```

```
;  3962          4083  1 GLOBAL ROUTINE DBG$ACTIVATE_EVENTS: NOVALUE =
;  3963          4084  1 !
;  3964          4085  1 ! FUNCTION
;  3965          4086  1 !       This routine activates all set events.
;  3966          4087  1 !
;  3967          4088  1 ! INPUTS
;  3968          4089  1 !       NONE
;  3969          4090  1 !
;  3970          4091  1 ! OUTPUTS
;  3971          4092  1 !       The events are activated, including the setting of breakpoints, the
;  3972          4093  1 !       setting of protections, etc.  The exception lists are also built.
;  3973          4094  1 !
;  3974          4095  1
;  3975          4096  2     BEGIN
;  3976          4097  2
;  3977          4098  2     LOCAL
;  3978          4099  2         USERS_FP: REF BLOCK [,BYTE],
;  3979          4100  2         EVENT_ENTRY :   REF EVENT$EVENT_DESCRIPTOR, ! Event entry pointer
;  3980          4101  2         NEXT_ENTRY :    REF EVENT$EVENT_DESCRIPTOR, ! Event entry pointer
;  3981          4102  2         PAGE_ENTRY :    REF EVENT$PAGE_DESCRIPTOR;  ! Page entry pointer
;  3982          4103  2
;  3983          4104  2
;  3984          4105  2     ! Turn off the TBIT flag, which will be set by any event(s) that need
;  3985          4106  2     ! to use it....
;  3986          4107  2     !
;  3987          4108  2     DBG$RUNFRAME [DBG$V_TBIT] = FALSE;
;  3988          4109  2
;  3989          4110  2
;  3990          4111  2     ! Point to the first Command Queue entry, and begin to process each entry
;  3991          4112  2     ! in turn.
;  3992          4113  2     !
;  3993          4114  2     USERS_FP = 0;
;  3994          4115  2     EVENT_ENTRY = .EVENT$CMD_QUEUE [L_QUEUE_FLINK];
;  3995          4116  2     WHILE .EVENT_ENTRY NEQA EVENT$CMD_QUEUE [A_QUEUE_ENTRY] DO
;  3996          4117  3         BEGIN
;  3997          4118  3
;  3998          4119  3
;  3999          4120  3         ! Point to the next entry (in case we end up deleting this one !!!
;  4000          4121  3         !
;  4001          4122  3         NEXT_ENTRY = .EVENT_ENTRY [EVENT$L_CMD_FLINK];
;  4002          4123  3
;  4003          4124  3
;  4004          4125  3         ! 'Delete' IGNOR entries....
;  4005          4126  3         !
;  4006          4127  3         IF .EVENT_ENTRY [EVENT$B_CMD_TYPE] EQLU EVENT$K_TYPE_IGNOR
;  4007          4128  3         THEN
;  4008          4129  3             EVENT_ENTRY [EVENT$V_DELETED] = TRUE;
;  4009          4130  3
;  4010          4131  3
;  4011          4132  3         ! Delete or Activate this event.
;  4012          4133  3         !
;  4013          4134  3         IF .EVENT_ENTRY [EVENT$V_DELETED]
;  4014          4135  3         THEN
;  4015          4136  4             BEGIN
;  4016          4137  4
;  4017          4138  4             ! *****SSI
;  4018          4139  4             ! EVENT$K_TYPE_SSINT Event is put on Event List, when we see
```

```
;  4019          4140  4            ! DBG$_SS_INT signal comes in (DEBUG interception routine is
;  4020          4141  4            ! called, trigged by System Service Call).  Bit 15 is set in
;  4021          4142  4            ! saved PSW in the FP for System service.  When system service
;  4022          4143  4            ! returns, DEBUG catches Reserved Operand Fault, EVENT$K_TYPE_SSINT
;  4023          4144  4            ! is marked deleted.  Now, we just about to take this event entry
;  4024          4145  4            ! off the Event List.
;  4025          4146  4            ! Check to see if we just got back from Reserved Operand Fault
;  4026          4147  4            ! from intercept system service RET, if it is, remember FP.
;  4027          4148  4            ! For we need to turn S/RET active entry to delete entry.
;  4028          4149  4            ! S/RET was built from step over the CALLS SYS$XXX entry, since
;  4029          4150  4            ! we already got Reserved Operand Fault from RET, so we don't
;  4030          4151  4            ! want to do it again, so later on we can turn S/RET into normal
;  4031          4152  4            ! step entry.
;  4032          4153  4            !
;  4033          4154  4            IF .EVENT_ENTRY [EVENT$B_CMD_TYPE] EQL EVENT$K_TYPE_SSINT
;  4034          4155  4            THEN
;  4035          4156  4                USERS_FP = .EVENT_ENTRY [EVENT$L_USERS_FP]
;  4036          4157  4            ELSE
;  4037          4158  4                USERS_FP = 0;
;  4038          4159  4            DELETE_EVENT_ENTRY (.EVENT_ENTRY);
;  4039          4160  4            END
;  4040          4161  3        ELSE
;  4041          4162  3            DBG$ACTIVATE_EVENT (.EVENT_ENTRY);
;  4042          4163  3
;  4043          4164  3
;  4044          4165  3        ! Point to the next event and continue.
;  4045          4166  3        !
;  4046          4167  3        EVENT_ENTRY = .NEXT_ENTRY;
;  4047          4168  3
;  4048          4169  3
;  4049          4170  3        ! *****SSI
;  4050          4171  3        ! Check to see if this the case we noted above.  Turn S/RET into
;  4051          4172  3        ! normal Step entry.
;  4052          4173  3        !
;  4053          4174  3        IF .USERS_FP NEQ 0
;  4054          4175  3        THEN
;  4055          4176  4            BEGIN
;  4056          4177  4            IF .EVENT_ENTRY [EVENT$B_SUB_KIND] EQL EVENT$K_ACC_RTRN
;  4057          4178  4            THEN
;  4058          4179  5                BEGIN
;  4059          4180  5                IF .EVENT_ENTRY [EVENT$B_CMD_TYPE] EQL EVENT$K_TYPE_STEPS
;  4060          4181  5                THEN
;  4061          4182  5                    IF .USERS_FP EQL .EVENT_ENTRY [EVENT$L_USERS_FP]
;  4062          4183  5                    THEN
;  4063          4184  5
;  4064          4185  5
;  4065          4186  5                        ! Change it to regular step entry.
;  4066          4187  5                        !
;  4067          4188  6                        BEGIN
;  4068          4189  6                        IF .EVENT_ENTRY [EVENT$V_STEP_BPT]
;  4069          4190  6                        THEN
;  4070          4191  7                            BEGIN
;  4071          4192  7                            EVENT_ENTRY [EVENT$V_STEP_BPT] = FALSE;
;  4072          4193  7                            EVENT_ENTRY [EVENT$B_CMD_KIND] = EVENT$K_KIND_INS;
;  4073          4194  7                            EVENT_ENTRY [EVENT$B_SUB_KIND] = .EVENT_ENTRY [EVENT$B_SAVED_SUB_KIND];
;  4074          4195  6                            END;
;  4075          4196  5                        END;
```

```
;  4076          4197  4                  END;
;  4077          4198  3              END;
;  4078          4199  2          END;
;  4079          4200  2
;  4080          4201  2
;  4081          4202  2          ! As long as were not skipping ACCVIO's, for each entry in the page
;  4082          4203  2          ! queue, write protect the page described.  If we ARE skipping then
;  4083          4204  2          ! we have to TBIT.
;  4084          4205  2          ! *****SSI
;  4085          4206  2          ! If we are in the middle of the system service cycle, we don't want
;  4086          4207  2          ! want to put the protection back, for if we do, we may cause the
;  4087          4208  2          ! system service to fail again.  Note: Some system service
;  4088          4209  2          ! calls system service, the cycle starts from the call in user program
;  4089          4210  2          ! till this call is returned to the user program.  During this time
;  4090          4211  2          ! frame, we don't want to put protection back.
;  4091          4212  2          !
;  4092          4213  2          IF NOT .SKIP_ACCVIOS AND
;  4093          4214  2             NOT .DBG$GB_SET_WATCH_FLAG
;  4094          4215  2          THEN
;  4095          4216  3              BEGIN
;  4096          4217  3              PAGE_ENTRY = .EVENT$PAGE_QUEUE [L_QUEUE_FLINK];
;  4097          4218  3              WHILE .PAGE_ENTRY NEQA EVENT$PAGE_QUEUE DO
;  4098          4219  4                  BEGIN
;  4099          4220  4                  LOCAL
;  4100          4221  4                      ADDRESS :       VECTOR [2];     ! Address range for protect
;  4101          4222  4
;  4102          4223  4                  ADDRESS [0] = .PAGE_ENTRY [EVENT$L_PAGE_ADDRESS];
;  4103          4224  4                  ADDRESS [1] = .PAGE_ENTRY [EVENT$L_PAGE_ADDRESS];
;  4104        P 4225  4                  IF NOT $SETPRT (INADR       = ADDRESS,
;  4105        P 4226  4                                  PROT        = PRT$C_UR,
;  4106        P 4227  4                                  PRVPRT      = PAGE_ENTRY [EVENT$B_PAGE_PROTECTION]
;  4107          4228  5                                 )
;  4108          4229  4                  THEN
;  4109          4230  4                      SIGNAL (DBG$_NOWPROT)
;  4110          4231  4                  ELSE
;  4111          4232  4                      PAGE_ENTRY = .PAGE_ENTRY [EVENT$L_PAGE_FLINK];
;  4112          4233  3                  END;
;  4113          4234  2              END;
;  4114          4235  2
;  4115          4236  2
;  4116          4237  2          ! If we are skipping ACCVIO and WATCHES, T-bit over the cause.
;  4117          4238  2          ! (just had a accvio from the instruction write to the page that was
;  4118          4239  2          !  protected for watch pointing) or (just had a RET from Systerm Service).
;  4119          4240  2          !
;  4120          4241  2          IF .SKIP_ACCVIOS OR .SKIP_WATCHES
;  4121          4242  2          THEN
;  4122          4243  3              BEGIN
;  4123          4244  3
;  4124          4245  3              ! About to TBIT over the instruction that triggered a watchpoint.
;  4125          4246  3              ! We also have to make sure that we don't get interrupted by ASTs.
;  4126          4247  3              !
;  4127          4248  3              DBG$RUNFRAME [DBG$V_TBIT] = TRUE;
;  4128          4249  3              DBG$GV_CONTROL[DBG$V_CONTROL_TBIT] = TRUE;
;  4129          4250  3              END;
;  4130          4251  2
;  4131          4252  1      END;
```

```
                                00FC 00000          .ENTRY  DBG$ACTIVATE_EVENTS, Save R2,R3,R4,R5,R6,R7 ; 4083
                    57 00000000G   00   9E 00002    MOVAB   DBG$RUNFRAME+68, R7                         ;
                    56 00000000'   EF   9E 00009    MOVAB   SKIP_ACCVIOS, R6                            ;
                    55 00000000'   EF   9E 00010    MOVAB   EVENT$CMD_QUEUE, R5                         ;
                    5E             08   C2 00017    SUBL2   #8, SP                                      ;
                    67             10   8A 0001A    BICB2   #16, DBG$RUNFRAME+68                        ; 4108
                                   53   D4 0001D    CLRL    USERS_FP                                    ; 4114
                    52             65   D0 0001F    MOVL    EVENT$CMD_QUEUE, EVENT_ENTRY                ; 4115
                    50             65   9E 00022 1$: MOVAB   EVENT$CMD_QUEUE, R0                        ; 4116
                    50             52   D1 00025    CMPL    EVENT_ENTRY, R0                             ;
                                   5E   13 00028    BEQL    7$                                          ;
                    54             62   D0 0002A    MOVL    (EVENT_ENTRY), NEXT_ENTRY                   ; 4122
                    05       14    A2   91 0002D    CMPB    20(EVENT_ENTRY), #5                         ; 4127
                                   05   12 00031    BNEQ    2$                                          ;
              17    A2       80    8F   88 00033    BISB2   #128, 23(EVENT_ENTRY)                       ; 4129
                             17    A2   95 00038 2$: TSTB    23(EVENT_ENTRY)                             ; 4134
                                   17   18 0003B    BGEQ    5$                                          ;
                    06       14    A2   91 0003D    CMPB    20(EVENT_ENTRY), #6                         ; 4154
                                   06   12 00041    BNEQ    3$                                          ;
                    53       24    A2   D0 00043    MOVL    36(EVENT_ENTRY), USERS_FP                   ; 4156
                                   02   11 00047    BRB     4$                                          ;
                                   53   D4 00049 3$: CLRL    USERS_FP                                    ; 4158
                                   52   DD 0004B 4$: PUSHL   EVENT_ENTRY                                ; 4159
              0000V CF             01   FB 0004D    CALLS   #1, DELETE_EVENT_ENTRY                      ;
                                   07   11 00052    BRB     6$                                          ; 4134
                                   52   DD 00054 5$: PUSHL   EVENT_ENTRY                                ; 4162
              0000V CF             01   FB 00056    CALLS   #1, DBG$ACTIVATE_EVENT                      ;
                    52             54   D0 0005B 6$: MOVL    NEXT_ENTRY, EVENT_ENTRY                     ; 4167
                                   53   D5 0005E    TSTL    USERS_FP                                    ; 4174
                                   C0   13 00060    BEQL    1$                                          ;
                    04       16    A2   91 00062    CMPB    22(EVENT_ENTRY), #4                         ; 4177
                                   BA   12 00066    BNEQ    1$                                          ;
                    03       14    A2   91 00068    CMPB    20(EVENT_ENTRY), #3                         ; 4180
                                   B4   12 0006C    BNEQ    1$                                          ;
              24    A2             53   D1 0006E    CMPL    USERS_FP, 36(EVENT_ENTRY)                   ; 4182
                                   AE   12 00072    BNEQ    1$                                          ;
         A9   19    A2             02   E1 00074    BBC     #2, 25(EVENT_ENTRY), 1$                     ; 4189
              19    A2             04   8A 00079    BICB2   #4, 25(EVENT_ENTRY)                         ; 4192
              15    A2             01   90 0007D    MOVB    #1, 21(EVENT_ENTRY)                         ; 4193
              16    A2       1B    A2   90 00081    MOVB    27(EVENT_ENTRY), 22(EVENT_ENTRY)            ; 4194
                                   9A   11 00086    BRB     1$                                          ; 4116
                    49             66   E8 00088 7$: BLBS    SKIP_ACCVIOS, 11$                           ; 4213
                    3E       FD    A5   E8 0008B    BLBS    DBG$GB_SET_WATCH_FLAG, 10$                  ; 4214
                    52       30    A5   D0 0008F    MOVL    EVENT$PAGE_QUEUE, PAGE_ENTRY                ; 4217
                    50       30    A5   9E 00093 8$: MOVAB   EVENT$PAGE_QUEUE, R0                       ; 4218
                    50             52   D1 00097    CMPL    PAGE_ENTRY, R0                              ;
                                   31   13 0009A    BEQL    10$                                         ;
                    6E       08    A2   D0 0009C    MOVL    8(PAGE_ENTRY), ADDRESS                      ; 4223
              04    AE       08    A2   D0 000A0    MOVL    8(PAGE_ENTRY), ADDRESS+4                    ; 4224
                             0E    A2   9F 000A5    PUSHAB  14(PAGE_ENTRY)                              ; 4228
                                   0F   DD 000A8    PUSHL   #15                                         ;
                                   7E   7C 000AA    CLRQ    -(SP)                                       ;
                             10    AE   9F 000AC    PUSHAB  ADDRESS                                     ;
```

```
                          00000000G  00              05  FB 000AF          CALLS   #5, SYS$SETPRT                          ;
                                     0F              50  E8 000B6          BLBS    R0, 9$                                 ;
                                           000284C4  8F  DD 000B9          PUSHL   #165060                                ;  4230
                          00000000G  00              01  FB 000BF          CALLS   #1, LIB$SIGNAL                         ;
                                     52              CB  11 000C6          BRB     8$                                     ;
                                                     62  D0 000C8  9$:      MOVL    (PAGE_ENTRY), PAGE_ENTRY               ;  4232
                                     04              C6  11 000CB          BRB     8$                                     ;  4218
                                     0A              66  E8 000CD  10$:     BLBS    SKIP_ACCVIOS, 11$                      ;  4241
                                     67        04    A6  E9 000D0          BLBC    SKIP_WATCHES, 12$                      ;
                          00000000G  00              10  88 000D4  11$:     BISB2   #16, DBG$RUNFRAME+68                   ;  4248
                                                     04  88 000D7          BISB2   #4, DBG$GV_CONTROL+1                   ;  4249
                                                         04 000DE  12$:     RET                                            ;  4252

; Routine Size:  223 bytes,    Routine Base:  DBG$CODE + 196D
```

```
;  4133          4253  1 GLOBAL ROUTINE DBG$ACTIVATE_EVENT (EVENT): NOVALUE =
;  4134          4254  1 !
;  4135          4255  1 ! FUNCTION
;  4136          4256  1 !       This routine activates an event.  Included in this action is the
;  4137          4257  1 !       building of a linked list for each exception type used to implement
;  4138          4258  1 !       any event:  BPT, TBIT, and ACCVIO.
;  4139          4259  1 !
;  4140          4260  1 ! INPUTS
;  4141          4261  1 !       EVENT :         Pointer to an event entry.
;  4142          4262  1 !
;  4143          4263  1 ! OUTPUTS
;  4144          4264  1 !       The event is activated, including the setting of breakpoints, the
;  4145          4265  1 !       setting of protections, etc.  The exception lists are also built.
;  4146          4266  1 !
;  4147          4267  1
;  4148          4268  2     BEGIN
;  4149          4269  2     MAP EVENT :             REF EVENT$EVENT_DESCRIPTOR; ! Event entry pointer
;  4150          4270  2     BUILTIN PROBER;
;  4151          4271  2
;  4152          4272  2     LOCAL
;  4153          4273  2         ADDRESS,
;  4154          4274  2         AT_LIBSIGNAL,                                   ! Flag
;  4155          4275  2         CALL_FRAME: REF VECTOR[,LONG],                  ! Save part of the FP
;  4156          4276  2         IGNOR_ENTRY :   REF EVENT$EVENT_DESCRIPTOR,     ! Event entry pointer
;  4157          4277  2         EVENT_ADDRESS : REF VECTOR [, BYTE],            ! Event byte address
;  4158          4278  2         USERS_FP :      REF BLOCK [, BYTE],             ! User's current FP
;  4159          4279  2         RET_PC,                                         ! Return PC
;  4160          4280  2         STATUS,                                         ! Status value
;  4161          4281  2         USERS_PC: REF VECTOR[,BYTE];                    ! Users current PC
;  4162          4282  2
;  4163          4283  2     ! Case on the kind of event entry, to determine whether how to activate
;  4164          4284  2     ! the entry.
;  4165          4285  2     !
;  4166          4286  2     CASE .EVENT [EVENT$B_CMD_KIND]
;  4167          4287  2     FROM EVENT$K_KIND_ACC TO EVENT$K_KIND_EXC OF
;  4168          4288  2         SET
;  4169          4289  2         [EVENT$K_KIND_ACC]:
;  4170          4290  2
;  4171          4291  2
;  4172          4292  2             ! Case on the access sub-kind....
;  4173          4293  2             !
;  4174          4294  2             CASE .EVENT[EVENT$B_SUB_KIND]
;  4175          4295  2             FROM EVENT$K_ACC_READ TO EVENT$K_ACC_RTRN OF
;  4176          4296  2                 SET
;  4177          4297  2
;  4178          4298  2
;  4179          4299  2                 [EVENT$K_ACC_READ]:
;  4180          4300  2                     ;
;  4181          4301  2
;  4182          4302  2
;  4183          4303  2                 [EVENT$K_ACC_WRIT]:
;  4184          4304  2                     ;
;  4185          4305  2
;  4186          4306  2
;  4187          4307  2                 [EVENT$K_ACC_MDFY]:
;  4188          4308  2                     ;
;  4189          4309  2
```

```
        ! Execution Access means that we've got to put a BPT
        ! instruction at the address indicated.
        !
        [EVENT$K_ACC_EXEC]:
            BEGIN

            ! Get the event's address.
            !
            EVENT_ADDRESS = .EVENT [EVENT$L_ADDRESS];

            ! Next, copy the opcode at the event's address into the
            ! event entry for safe keeping.
            !
            EVENT [EVENT$B_OPCODE] = .EVENT_ADDRESS [0];

            ! If the address we're to place a BPT is the current user
            ! address, (meaning we're about to put a BPT where we're
            ! about to start), create a ignore-event instead....
            !
            IF .EVENT_ADDRESS EQLA .DBG$RUNFRAME[DBG$L_USER_PC]
            THEN
                BEGIN

                ! Get a new event descriptor, and link it into the
                ! command queue.  Note that this entry is placed on
                ! the HEAD of the queue, and should be encountered
                ! first !!!
                !
                IGNOR_ENTRY = DBG$GET_MEMORY (EVENT$K_EVENT_DESCRIPTOR_SIZE);
                INSQUE (.IGNOR_ENTRY, EVENT$CMD_QUEUE);

                ! Setup this entry to be a ignore-event.  Note that
                ! we don't really need to initialize more than the
                ! CMD_TYPE, including linking to the event entry
                ! causing this mess, 'cause this entry will just be
                ! deleted.  However, maybe in the future....
                !
                IGNOR_ENTRY [EVENT$B_CMD_TYPE] = EVENT$K_TYPE_IGNOR;
                IGNOR_ENTRY [EVENT$B_CMD_KIND] = EVENT$K_KIND_ACC;
                IGNOR_ENTRY [EVENT$B_SUB_KIND] = EVENT$K_ACC_EXEC;
                IGNOR_ENTRY [EVENT$V_ONCE_ONLY] = TRUE;
                INSQUE1 (.IGNOR_ENTRY, .EVENT);

                ! Turn on the TBIT flag so that we'll TBIT over the
                ! next instruction, and return.
                !
                DBG$RUNFRAME [DBG$V_TBIT] = TRUE;
                DBG$GV_CONTROL[DBG$V_CONTROL_TBIT] = TRUE;
                RETURN;
                END;
```

```
;  4247          4367  3
;  4248          4368  3
;  4249          4369  3                    ! BPT is okay, write a BPT instruction into that address.
;  4250          4370  3                    ! Note that we won't actually do this if the original
;  4251          4371  3                    ! opcode was a BPT.
;  4252          4372  3                    !
;  4253          4373  3                    IF .EVENT [EVENT$B_OPCODE] NEQU BPT_INSTRUCTION
;  4254          4374  3                    THEN
;  4255          4375  4                        IF NOT WRITE_OPCODE (.EVENT_ADDRESS, BPT_INSTRUCTION)
;  4256          4376  4                        THEN
;  4257          4377  4                            BEGIN
;  4258          4378  4                            SIGNAL (DBG$_NOWBPT);
;  4259          4379  4                            RETURN;
;  4260          4380  4                            END;
;  4261          4381  3                    END;
;  4262          4382  3
;  4263          4383  3
;  4264          4384  2                ! Return Access means that we've got to put a BPT
;  4265          4385  2                ! instruction at the address indicated.
;  4266          4386  2                !
;  4267          4387  2                [EVENT$K_ACC_RTRN]:
;  4268          4388  2                    BEGIN
;  4269          4389  2
;  4270          4390  2
;  4271          4391  2                    ! Depending on the command type (STEP or otherwise)...
;  4272          4392  2                    !
;  4273          4393  2                    IF .EVENT [EVENT$B_CMD_TYPE] EQLU EVENT$K_TYPE_STEPS OR
;  4274          4394  2                       .EVENT [EVENT$B_CMD_TYPE] EQLU EVENT$K_TYPE_SKIPS OR
;  4275          4395  2                       .EVENT [EVENT$B_CMD_TYPE] EQLU EVENT$K_TYPE_SSINT
;  4276          4396  2                    THEN
;  4277          4397  3                        BEGIN
;  4278          4398  3
;  4279          4399  3
;  4280          4400  3                        ! If the save FP is 0, then we need to initialize
;  4281          4401  3                        ! with the current one.
;  4282          4402  3                        !
;  4283          4403  3                        USERS_FP = .EVENT [EVENT$L_USERS_FP];
;  4284          4404  3                        IF .USERS_FP EQLA 0
;  4285          4405  3                        THEN
;  4286          4406  4                            BEGIN
;  4287          4407  4                            USERS_FP = .DBG$RUNFRAME [DBG$L_USER_FP];
;  4288          4408  4                            EVENT [EVENT$L_USERS_FP] = .USERS_FP;
;  4289          4409  4                            CALL_FRAME = DBG$GET_MEMORY(5);
;  4290          4410  4                            EVENT [EVENT$L_CALL_FRAME] = .CALL_FRAME;
;  4291          4411  4                            CH$MOVE(20,.EVENT[EVENT$L_USERS_FP],.CALL_FRAME);
;  4292          4412  3                            END;
;  4293          4413  3
;  4294          4414  3                        IF NOT PROBER (%REF(0), %REF(20), .USERS_FP)
;  4295          4415  3                        THEN
;  4296          4416  3                            RETURN;
;  4297          4417  3
;  4298          4418  3
;  4299          4419  3                        ! Check for the special case where we have just
;  4300          4420  3                        ! called LIB$SIGNAL or LIB$STOP. The flag
;  4301          4421  3                        ! AT_LIBSIGNAL will be set to true if we are
;  4302          4422  3                        ! at LIB$SIGNAL or LIB$STOP
;  4303          4423  3                        !
```

```
;  4304          4424  4                    USERS_PC = .DBG$RUNFRAME[DBG$L_USER_PC];
;  4305          4425  4                    AT_LIBSIGNAL = FALSE;
;  4306          4426  4
;  4307          4427  4
;  4308          4428  4                    ! Check for current PC being at one of these
;  4309          4429  4                    ! locations. Here the locations we are checking
;  4310          4430  4                    ! are the addresses of LIB$SIGNAL and LIB$STOP
;  4311          4431  4                    ! in the LIBRTL shareable library.
;  4312          4432  4                    !
;  4313          4433  4                    IF (.USERS_PC EQL LIB$SIGNAL+2) OR
;  4314          4434  5                       (.USERS_PC EQL LIB$STOP+2)
;  4315          4435  4                    THEN
;  4316          4436  4                        AT_LIBSIGNAL = TRUE;
;  4317          4437  4
;  4318          4438  4
;  4319          4439  4                    ! Also check for the current PC being at the
;  4320          4440  4                    ! address of LIB$SIGNAL or LIB$STOP if these
;  4321          4441  4                    ! were linked into the users program with
;  4322          4442  4                    ! /NOSYSSHR on the link. In this case the
;  4323          4443  4                    ! symbols show up in the GST and we remember
;  4324          4444  4                    ! their addresses when we initialize the GST.
;  4325          4445  4                    !
;  4326          4446  4                    IF (.USERS_PC EQL .DBG$GL_LIB_SIGNAL_ADDR+2) OR
;  4327          4447  5                       (.USERS_PC EQL .DBG$GL_LIB_STOP_ADDR+2)
;  4328          4448  4                    THEN
;  4329          4449  4                        AT_LIBSIGNAL = TRUE;
;  4330          4450  4
;  4331          4451  4
;  4332          4452  4                    ! Also check for instruction at current location
;  4333          4453  4                    ! being a JMP to LIB$SIGNAL or LIB$STOP.
;  4334          4454  4                    !
;  4335          4455  4                    IF PROBER(%REF(0),
;  4336          4456  4                              %REF(6),
;  4337          4457  4                              .USERS_PC)
;  4338          4458  4                    THEN
;  4339          4459  5                        BEGIN
;  4340          4460  5                        IF (.USERS_PC[0] EQL %X'17') AND
;  4341          4461  6                           (.USERS_PC[1] EQL %X'FF')
;  4342          4462  5                        THEN
;  4343          4463  6                            BEGIN
;  4344          4464  6
;  4345          4465  6                            ! Get the jump address by adding the PC
;  4346          4466  6                            ! value plus the 6 bytes for the
;  4347          4467  6                            ! JMP instruction itself.
;  4348          4468  6                            !
;  4349          4469  6                            CH$MOVE(4, USERS_PC[2], ADDRESS);
;  4350          4470  6                            ADDRESS = .ADDRESS + .USERS_PC + 6;
;  4351          4471  6                            IF PROBER(%REF(0),
;  4352          4472  6                                      %REF(4),
;  4353          4473  6                                      .ADDRESS)
;  4354          4474  6                            THEN
;  4355          4475  6                                ADDRESS = ..ADDRESS;
;  4356          4476  6
;  4357          4477  6                            IF (.ADDRESS EQL LIB$SIGNAL+2) OR
;  4358          4478  7                               (.ADDRESS EQL LIB$STOP+2)
;  4359          4479  6                            THEN
;  4360          4480  6                                AT_LIBSIGNAL = TRUE;
```

```
;  4361          4481  5                    END;
;  4362          4482  4                END;
;  4363          4483  4
;  4364          4484  4            IF .AT_LIBSIGNAL
;  4365          4485  4            THEN
;  4366          4486  5                BEGIN
;  4367          4487  5
;  4368          4488  5                ! In this case, we do not want want to use
;  4369          4489  5                ! the dirty-the-tbit approach because
;  4370          4490  5                ! LIB$SIGNAL and LIB$STOP mess with the stack.
;  4371          4491  5                ! So we revert to the temporary breakpoint
;  4372          4492  5                ! method.
;  4373          4493  5                !
;  4374          4494  5                RET_PC = .USERS_FP[SF$L_SAVE_PC];
;  4375          4495  5                EVENT [EVENT$V_STEP_BPT] = TRUE;
;  4376          4496  5                EVENT [EVENT$B_CMD_KIND] = EVENT$K_KIND_INS;
;  4377          4497  5                EVENT [EVENT$B_SUB_KIND] = .EVENT [EVENT$B_SAVED_SUB_KIND];
;  4378          4498  5                IF .EVENT [EVENT$B_SUB_KIND] EQL 0
;  4379          4499  5                THEN
;  4380          4500  5                    EVENT [EVENT$B_SUB_KIND] = EVENT$K_INS_EVRY;
;  4381          4501  5                EVENT [EVENT$L_ADDRESS] = .RET_PC;
;  4382          4502  5                EVENT [EVENT$B_OPCODE] = .(.RET_PC)<0,8>;
;  4383          4503  6                IF NOT WRITE_OPCODE (.RET_PC, BPT_INSTRUCTION)
;  4384          4504  5                THEN
;  4385          4505  5                    SIGNAL(DBG$_NOWBPT);
;  4386          4506  5                RETURN;
;  4387          4507  4                END;
;  4388          4508  4
;  4389          4509  4
;  4390          4510  4                ! Turn on bit 15 in the user's run-
;  4391          4511  4                ! frame to get a reserved-operand
;  4392          4512  4                ! fault on the return.
;  4393          4513  4                !
;  4394          4514  4                MATCH_RIGHT_CALL_FRAME(.EVENT);
;  4395          4515  4                USERS_FP = .EVENT [EVENT$L_USERS_FP];
;  4396          4516  4                (USERS_FP [SF$W_SAVE_PSW])<15,1,0> = TRUE;
;  4397          4517  4                END
;  4398          4518  4
;  4399          4519  4
;  4400          4520  4            ! This is a [BREAK!TRACE]/RETURN to set up.
;  4401          4521  4            !
;  4402          4522  3            ELSE
;  4403          4523  4                BEGIN
;  4404          4524  4
;  4405          4525  4
;  4406          4526  4                ! Get the event's address.
;  4407          4527  4                !
;  4408          4528  4                EVENT_ADDRESS = .EVENT [EVENT$L_ADDRESS];
;  4409          4529  4
;  4410          4530  4
;  4411          4531  4                ! Next, copy the opcode at the event's address into the
;  4412          4532  4                ! event entry for safe keeping.
;  4413          4533  4                !
;  4414          4534  4                EVENT [EVENT$B_OPCODE] = .EVENT_ADDRESS [0];
;  4415          4535  4
;  4416          4536  4
;  4417          4537  4                ! If the address we're to place a BPT is the current
```

```
;  4418          4538  4                ! user address, (meaning we're about to put a BPT
;  4419          4539  4                ! where we're about to start), create a ignore-event
;  4420          4540  4                ! instead....
;  4421          4541  4                !
;  4422          4542  4                IF .EVENT_ADDRESS EQLA .DBG$RUNFRAME[DBG$L_USER_PC]
;  4423          4543  4                THEN
;  4424          4544  5                    BEGIN
;  4425          4545  5
;  4426          4546  5                    ! Get a new event descriptor, and link it into the
;  4427          4547  5                    ! command queue.  Note that this entry is placed on
;  4428          4548  5                    ! the HEAD of the queue, and should be encountered
;  4429          4549  5                    ! first !!!
;  4430          4550  5                    !
;  4431          4551  5                    IGNOR_ENTRY = DBG$GET_MEMORY
;  4432          4552  5                                  (EVENT$K_EVENT_DESCRIPTOR_SIZE);
;  4433          4553  5                    INSQUE (.IGNOR_ENTRY, EVENT$CMD_QUEUE);
;  4434          4554  5                    !
;  4435          4555  5                    ! Setup this entry to be a ignore-event.  Note that
;  4436          4556  5                    ! we don't really need to initialize more than the
;  4437          4557  5                    ! CMD_TYPE, including linking to the event entry
;  4438          4558  5                    ! causing this mess, 'cause this entry will just be
;  4439          4559  5                    ! deleted.  However, maybe in the future....
;  4440          4560  5                    ! If we are at /RET .PC, then we setup a skip entry
;  4441          4561  5                    ! instead of ignore entry.
;  4442          4562  5                    !
;  4443          4563  5                    IF .EVENT [EVENT$V_RET_AT_PC]
;  4444          4564  5                    THEN
;  4445          4565  5                        IGNOR_ENTRY [EVENT$B_CMD_TYPE] = EVENT$K_TYPE_SKIPS
;  4446          4566  5                    ELSE
;  4447          4567  5                        IGNOR_ENTRY [EVENT$B_CMD_TYPE] = EVENT$K_TYPE_IGNOR;
;  4448          4568  5                    IGNOR_ENTRY [EVENT$B_CMD_KIND] = EVENT$K_KIND_ACC;
;  4449          4569  5                    IGNOR_ENTRY [EVENT$B_SUB_KIND] = EVENT$K_ACC_RTRN;
;  4450          4570  5                    IGNOR_ENTRY [EVENT$V_ONCE_ONLY] = TRUE;
;  4451          4571  5                    INSQUEI (.IGNOR_ENTRY, .EVENT);
;  4452          4572  5                    !
;  4453          4573  5                    ! Turn on the TBIT flag so that we'll TBIT over the
;  4454          4574  5                    ! next instruction, and return.
;  4455          4575  5                    !
;  4456          4576  5                    IF NOT .EVENT [EVENT$V_RET_AT_PC]
;  4457          4577  5                    THEN
;  4458          4578  5                        BEGIN
;  4459          4579  5                        DBG$RUNFRAME [DBG$V_TBIT] = TRUE;
;  4460          4580  5                        DBG$GV_CONTROL[DBG$V_CONTROL_TBIT] = TRUE;
;  4461          4581  5                        END;
;  4462          4582  6                    RETURN;
;  4463          4583  6                    END;
;  4464          4584  6
;  4465          4585  5                ! Write a BPT instruction into that address.  Note that
;  4466          4586  5                ! we won't actually do this if the original opcode was
;  4467          4587  4                ! a BPT.
;  4468          4588  4                !
;  4469          4589  4                IF .EVENT [EVENT$B_OPCODE] NEQU BPT_INSTRUCTION
;  4470          4590  4
;  4471          4591  4
;  4472          4592  4
;  4473          4593  4
;  4474          4594  4
```

```
;  4475     4595  4                    THEN
;  4476     4596  5                        IF NOT WRITE_OPCODE (.EVENT_ADDRESS, BPT_INSTRUCTION)
;  4477     4597  4                        THEN
;  4478     4598  5                            BEGIN
;  4479     4599  5                            SIGNAL (DBG$_NOWBPT);
;  4480     4600  5                            RETURN;
;  4481     4601  4                            END;
;  4482     4602  3                    END;
;  4483     4603  2                END;
;  4484     4604  2            TES;
;  4485     4605  2
;  4486     4606  2
;  4487     4607  2        [EVENT$K_KIND_INS]:
;  4488     4608  3            BEGIN
;  4489     4609  3            LOCAL OPCODE;
;  4490     4610  3
;  4491     4611  3            OPCODE = .(.DBG$RUNFRAME[DBG$L_USER_PC])<0,8,0>;
;  4492     4612  3            ! If stepping over and a) it's time to BPT or b) we WERE BPTing...
;  4493     4613  3            !
;  4494     4614  3            IF .EVENT [EVENT$V_STEP_OVER] AND
;  4495     4615  4            (.EVENT [EVENT$V_STEP_BPT] OR .DBG$OPCODES_STEP_OVER[.OPCODE])
;  4496     4616  3            THEN
;  4497     4617  4                BEGIN
;  4498     4618  4
;  4499     4619  4                ! ... and we're not already in the middle...
;  4500     4620  4                !
;  4501     4621  4                IF NOT .EVENT [EVENT$V_STEP_BPT]
;  4502     4622  4                THEN
;  4503     4623  5                    BEGIN
;  4504     4624  6                    IF (.OPCODE EQL %X'FA') OR (.OPCODE EQL %X'FB')
;  4505     4625  5                    THEN
;  4506     4626  6                        BEGIN
;  4507     4627  6
;  4508     4628  6                        ! This is a CALLS or CALLG instruction. We implement
;  4509     4629  6                        ! STEP/OVER by a STEP/INTO,STEP/RETURN,STEP sequence
;  4510     4630  6                        ! We do this by temporarily turning the STEP event descriptor into a RETURN
;  4511     4631  6                        ! event descriptor (KIND = ACC, SUB_KIND = RTRN). This causes us to dirty up
;  4512     4632  6                        ! the call stack and get a reserved operand fault at the return. When we handle the
;  4513     4633  6                        ! ROPRAND then we change the return event back into a step event. We save the origin
;  4514     4634  6                        ! sub-kind field to enable us to do this correctly.
;  4515     4635  6                        !
;  4516     4636  6                        EVENT [EVENT$V_STEP_BPT] = TRUE;
;  4517     4637  6                        EVENT [EVENT$L_USER$_FP] = 0;
;  4518     4638  6                        EVENT [EVENT$B_CMD_KIND] = EVENT$K_KIND_ACC;
;  4519     4639  6                        EVENT [EVENT$B_SAVED_SUB_KIND] = .EVENT [EVENT$B_SUB_KIND];
;  4520     4640  6                        EVENT [EVENT$B_SUB_KIND] = EVENT$K_ACC_RTRN;
;  4521     4641  6                        DBG$RUNFRAME [DBG$V_TBIT] = TRUE;
;  4522     4642  6                        DBG$GV_CONTROL[DBG$V_CONTROL_TBIT] = TRUE;
;  4523     4643  6                        RETURN;
;  4524     4644  6                        END
;  4525     4645  5                    ELSE
;  4526     4646  6                        BEGIN
;  4527     4647  6
;  4528     4648  6                        ! This is a JSB-type instruction.  Get the address
;  4529     4649  6                        ! of the next instruction.
;  4530     4650  6                        !
;  4531     4651  6                        EVENT_ADDRESS = DBG$INS_DECODE (
```

```
;  4532          4652  6                      .DBG$RUNFRAME[DBG$L_USER_PC], 0, FALSE);
;  4533          4653  6
;  4534          4654  6                  ! If the address is in system or P1 space...
;  4535          4655  6                  !
;  4536          4656  6                  IF .EVENT_ADDRESS GEQA P1_SPACE OR
;  4537          4657  6                     .EVENT_ADDRESS EQLA 0
;  4538          4658  6                  THEN
;  4539          4659  6
;  4540          4660  6
;  4541          4661  6                      ! ... then we're going to TBIT ('cause we won't set a
;  4542          4662  6                      ! BPT in P1 space or in system space).
;  4543          4663  6                      !
;  4544          4664  6                      BEGIN
;  4545          4665  7
;  4546          4666  7
;  4547          4667  7                      ! Check to see if this is address 0 or we're stepping
;  4548          4668  7                      ! nosystem.  If so - belch.
;  4549          4669  7                      !
;  4550          4670  7                      IF .EVENT [EVENT$V_STEP_NOSYSTEM] OR
;  4551          4671  7                         .EVENT_ADDRESS EQLA 0
;  4552          4672  7                      THEN
;  4553          4673  7                          SIGNAL (DBG$_STEPINTO, 1, .DBG$RUNFRAME[DBG$L_USER_PC]);
;  4554          4674  7
;  4555          4675  7
;  4556          4676  7                      ! We're TBIT'n now...
;  4557          4677  7                      !
;  4558          4678  7                      DBG$RUNFRAME [DBG$V_TBIT] = TRUE;
;  4559          4679  7                      RETURN;
;  4560          4680  7                      END;
;  4561          4681  6
;  4562          4682  6
;  4563          4683  6                  ! Otherwise, everything is hunky dory, so continue.  We'll
;  4564          4684  6                  ! Set up the address in the event entry for the BPT.
;  4565          4685  6                  !
;  4566          4686  6                  ! Save the next address in to entry for the step's BPT.
;  4567          4687  6                  !
;  4568          4688  6                  EVENT [EVENT$L_ADDRESS] = .EVENT_ADDRESS;
;  4569          4689  6
;  4570          4690  6
;  4571          4691  6                  ! Flag that a stepping-into BPT is set.
;  4572          4692  6                  !
;  4573          4693  6                  EVENT [EVENT$V_STEP_BPT] = TRUE;
;  4574          4694  6                  END;
;  4575          4695  5              END
;  4576          4696  5
;  4577          4697  5
;  4578          4698  5              ! We're in the middle of stepping over, but we've been
;  4579          4699  5              ! interrupted, so continue....
;  4580          4700  5              !
;  4581          4701  5              ELSE
;  4582          4702  4                  EVENT_ADDRESS = .EVENT [EVENT$L_ADDRESS];
;  4583          4703  4
;  4584          4704  4
;  4585          4705  4
;  4586          4706  4
;  4587          4707  4              ! We get here when a) we're in the middle of a step/over (i.e.
;  4588          4708  4              ! something interrupted the step/over), or we're at the start
```

```
4                    ! of one (i.e. at a CALL/JSB-type instruction), and we've got
4                    ! the address set up in the event entry (see above).
4                    !
4                    ! Next, copy the opcode at the event's address into the
4                    ! event entry for safe keeping.
4                    !
4                   EVENT [EVENT$B_OPCODE] = .EVENT_ADDRESS [0];
4
4
4                    ! Finally, write a BPT instruction into that address.
4                    ! Note that we won't actually do this if the original
4                    ! opcode was a BPT.
4                    !
4                   IF .EVENT [EVENT$B_OPCODE] NEQU BPT_INSTRUCTION
4                   THEN
5                       IF NOT WRITE_OPCODE (.EVENT_ADDRESS, BPT_INSTRUCTION)
4                       THEN
5                           BEGIN
5                           SIGNAL (DBG$_STEPINTO, 1, .DBG$RUNFRAME[DBG$L_USER_PC]);
5                           DBG$RUNFRAME [DBG$V_TBIT] = TRUE;
5                           EVENT [EVENT$V_STEP_BPT] = FALSE;
5                           RETURN;
4                           END;
4                   END
3               ELSE
3
3                    ! We're not stepping over, or we aren't going to BPT, so we'll
3                    ! TBIT.
3                    !
4                   BEGIN
4                   EVENT [EVENT$V_STEP_BPT] = FALSE;
4                   DBG$RUNFRAME [DBG$V_TBIT] = TRUE;
3                   END;
2               END;
2
2
2           [EVENT$K_KIND_EXC]:
2               ;
2           TES;
1       END;



                                        0FFC 00000        .ENTRY  DBG$ACTIVATE_EVENT, Save R2,R3,R4,R5,R6,R7,-; 4253
                                                                  R8,R9,R10,R11                                :
                  5B 00000000G   00  9E 00002        MOVAB   DBG$RUNFRAME+64, R11                         :
                  5E             14  C2 00009        SUBL2   #20, SP                                      :
                  56       04    AC  D0 0000C        MOVL    EVENT, R6                                    ; 4286
                  59       14    A6  9E 00010        MOVAB   20(R6), R9                                   :
         02       00       01    A9  8F 00014        CASEB   1(R9), #0, #2                                :
       02C4     01EE           0006     00019 1$:    .WORD   2$-1$,-                                      :
                                                             28$-1$,-                                     :
                                                             41$-1$                                       :
         04       00       02    A9  8F 0001F 2$:    CASEB   2(R9), #0, #4                                ; 4294
 000B  02B9     02B9           02B9     00024 3$:    .WORD   41$-3$,-                                     :
```

```
                              0066      0002C                41$-3$,-                                  :
                                                             41$-3$,-                                  :
                                                             4$-3$,-                                   :
                                                             8$-3$                                     :
                                   04   0002E       RET                                                :
               58        2C   A6   D0   0002F 4$:   MOVL     44(R6), EVENT_ADDRESS                     : 4320
         30    A6             68   90   00033       MOVB     (EVENT_ADDRESS), 48(R6)                   : 4326
               6B             58   D1   00037       CMPL     EVENT_ADDRESS, DBG$RUNFRAME+64            : 4333
                              2C   12   0003A       BNEQ     5$                                        :
                              10   DD   0003C       PUSHL    #16                                       : 4343
  00000000G    00             01   FB   0003E       CALLS    #1, DBG$GET_MEMORY                        :
               57             50   D0   00045       MOVL     R0, IGNOR_ENTRY                           :
  00000000'    EF             67   0E   00048       INSQUE   (IGNOR_ENTRY), EVENT$CMD_QUEUE            : 4344
         14    A7             05   B0   0004F       MOVW     #5, 20(IGNOR_ENTRY)                       : 4353
         16    A7             03   90   00053       MOVB     #3, 22(IGNOR_ENTRY)                       : 4355
         17    A7             04   88   00057       BISB2    #4, 23(IGNOR_ENTRY)                       : 4356
                         04   AC   DD   0005B       PUSHL    EVENT                                     : 4357
                              57   DD   0005E       PUSHL    IGNOR_ENTRY                               :
      0000V    CF             02   FB   00060       CALLS    #2, INSQUE1                               :
                            01E4   31   00065       BRW      32$                                       : 4363
               50        04   AC   D0   00068 5$:   MOVL     EVENT, R0                                 : 4373
               03        30   A0   91   0006C       CMPB     48(R0), #3                                :
                              4E   13   00070       BEQL     11$                                       :
               6E             03   D0   00072       MOVL     #3, OP                                    : 4375
                              01   DD   00075       PUSHL    #1                                        :
                         04   AE   9F   00077       PUSHAB   OP                                        :
                              58   DD   0007A       PUSHL    EVENT_ADDRESS                             :
  00000000G    00             03   FB   0007C 6$:   CALLS    #3, DBG$WRITE_MEM                         :
               03             50   E8   00083       BLBS     R0, 7$                                    :
                            0170   31   00086       BRW      27$                                       :
                                   04   00089 7$:   RET                                                : 4378
               03             69   91   0008A 8$:   CMPB     (R9), #3                                  : 4393
                              0D   13   0008D       BEQL     9$                                        :
               04             69   91   0008F       CMPB     (R9), #4                                  : 4394
                              08   13   00092       BEQL     9$                                        :
               06             69   91   00094       CMPB     (R9), #6                                  : 4395
                              03   13   00097       BEQL     9$                                        :
                            00EE   31   00099       BRW      22$                                       :
               5A        24   A6   D0   0009C 9$:   MOVL     36(R6), USERS_FP                          : 4403
                              1A   12   000A0       BNEQ     10$                                       : 4404
               5A        F8   AB   D0   000A2       MOVL     DBG$RUNFRAME+56, USERS_FP                 : 4407
         24    A6             5A   D0   000A6       MOVL     USERS_FP, 36(R6)                          : 4408
                              05   DD   000AA       PUSHL    #5                                        : 4409
  00000000G    00             01   FB   000AC       CALLS    #1, DBG$GET_MEMORY                        :
         3C    A6             50   D0   000B3       MOVL     CALL_FRAME, 60(R6)                        : 4410
60       24    B6             14   28   000B7       MOVC3    #20, @36(R6), (CALL_FRAME)                : 4411
6A             14             00   0C   000BC 10$:  PROBER   #0, #20, (USERS_FP)                       : 4414
                              01   12   000C0 11$:  BNEQ     12$                                       :
                                   04   000C2       RET                                                :
               50             6B   D0   000C3 12$:  MOVL     DBG$RUNFRAME+64, USERS_PC                 : 4424
                              52   D4   000C6       CLRL     AT_LIBSIGNAL                              : 4425
               51  00000000G  00   9E   000C8       MOVAB    LIB$SIGNAL+2, R1                          : 4433
               51             50   D1   000CF       CMPL     USERS_PC, R1                              :
                              0C   13   000D2       BEQL     13$                                       :
               51  00000000G  00   9E   000D4       MOVAB    LIB$STOP+2, R1                            : 4434
               51             50   D1   000DB       CMPL     USERS_PC, R1                              :
                              03   12   000DE       BNEQ     14$                                       :
```

```
                        52              01  D0 000E0 13$:   MOVL    #1, AT_LIBSIGNAL                        ; 4436
             51 00000000G   00          02  C1 000E3 14$:   ADDL3   #2, DBG$GL_LIB_SIGNAL_ADDR, R1          ; 4446
                        51              50  D1 000EB        CMPL    USERS_PC, R1
                                        0D  13 000EE        BEQL    15$
             51 00000000G   00          02  C1 000F0        ADDL3   #2, DBG$GL_LIB_STOP_ADDR, R1            ; 4447
                        51              50  D1 000F8        CMPL    USERS_PC, R1
                                        03  12 000FB        BNEQ    16$
                        52              01  D0 000FD 15$:   MOVL    #1, AT_LIBSIGNAL                        ; 4449
             60         06              00  0C 00100 16$:   PROBER  #0, #6, (USERS_PC)                      ; 4457
                                        43  13 00104        BEQL    19$
                        17              60  91 00106        CMPB    (USERS_PC), #23                         ; 4460
                                        3E  12 00109        BNEQ    19$
                   FF   8F      01      A0  91 0010B        CMPB    1(USERS_PC), #255                       ; 4461
                                        37  12 00110        BNEQ    19$
                   04   AE      02      A0  D0 00112        MOVL    2(USERS_PC), ADDRESS                    ; 4469
                   50   04      AE      C0  00117        ADDL2   ADDRESS, R0                             ; 4470
                   04   AE      06      A0  9E 0011B        MOVAB   6(R0), ADDRESS
          04   BE              04          00  0C 00120        PROBER  #0, #4, @ADDRESS                        ; 4473
                                        05  13 00125        BEQL    17$
                   04   AE      04      BE  D0 00127        MOVL    @ADDRESS, ADDRESS                       ; 4475
                   50 00000000G   00      9E 0012C 17$:   MOVAB   LIB$SIGNAL+2, R0                        ; 4477
                   50      04      AE      D1 00133        CMPL    ADDRESS, R0
                                        0D  13 00137        BEQL    18$
                   50 00000000G   00      9E 00139        MOVAB   LIB$STOP+2, R0                          ; 4478
                   50      04      AE      D1 00140        CMPL    ADDRESS, R0
                                        03  12 00144        BNEQ    19$
                        52              01  D0 00146 18$:   MOVL    #1, AT_LIBSIGNAL                        ; 4480
                        2D              52  E9 00149 19$:   BLBC    AT_LIBSIGNAL, 21$                       ; 4484
                   50      10      AA      D0 0014C        MOVL    16(USERS_FP), RET_PC                    ; 4494
                   19   A6              04  88 00150        BISB2   #4, 25(R6)                              ; 4495
                   01   A9              01  90 00154        MOVB    #1, 1(R9)                               ; 4496
                   02   A9      1B      A6  90 00158        MOVB    27(R6), 2(R9)                           ; 4497
                                        04  12 0015D        BNEQ    20$                                     ; 4498
                   02   A9              08  90 0015F        MOVB    #8, 2(R9)                               ; 4500
                   2C   A6              50  D0 00163 20$:   MOVL    RET_PC, 44(R6)                          ; 4501
                   30   A6              60  90 00167        MOVB    (RET_PC), 48(R6)                        ; 4502
                   08   AE              03  D0 0016B        MOVL    #3, OP                                  ; 4503
                                        01  DD 0016F        PUSHL   #1
                        0C      AE      9F  00171        PUSHAB  OP
                                        50  DD 00174        PUSHL   RET_PC
                                FF03    31  00176        BRW     6$
                                        56  DD 00179 21$:   PUSHL   R6                                     ; 4514
                0000V   CF              01  FB 0017B        CALLS   #1, MATCH_RIGHT_CALL_FRAME
                        5A      24      A6  D0 00180        MOVL    36(R6), USERS_FP                        ; 4515
                   05   AA      80      8F  88 00184        BISB2   #128, 5(USERS_FP)                       ; 4516
                                        04  00189        RET                                             ; 4393
                        58      2C      A6  D0 0018A 22$:   MOVL    44(R6), EVENT_ADDRESS                   ; 4528
                   30   A6              68  90 0018E        MOVB    (EVENT_ADDRESS), 48(R6)                 ; 4534
                        6B              58  D1 00192        CMPL    EVENT_ADDRESS, DBG$RUNFRAME+64          ; 4542
                                        41  12 00195        BNEQ    25$
                                        10  DD 00197        PUSHL   #16                                     ; 4553
             00000000G  00              01  FB 00199        CALLS   #1, DBG$GET_MEMORY
                        57              50  D0 001A0        MOVL    R0, IGNOR_ENTRY
             00000000'  EF              67  0E 001A3        INSQUE  (IGNOR_ENTRY), EVENT$CMD_QUEUE          ; 4554
                        50      14      A7  9E 001AA        MOVAB   20(IGNOR_ENTRY), R0                     ; 4567
                        52      04      AC  D0 001AE        MOVL    EVENT, R2                               ; 4565
          05        17      A2          05  E1 001B2        BBC     #5, 23(R2), 23$                         ;
```

```
                        60              04  90 001B7          MOVB    #4, (R0)                        ; 4567
                                        03  11 001BA          BRB     24$
                        60              05  90 001BC 23$:     MOVB    #5, (R0)                        ; 4569
                     01 A0      0400    8F  B0 001BF 24$:     MOVW    #1024, 1(R0)                    ; 4571
                     03 A0              04  88 001C5          BISB2   #4, 3(R0)                       ; 4573
                                        52  DD 001C9          PUSHL   R2                              ; 4574
                                        57  DD 001CB          PUSHL   IGNOR_ENTRY
                  0000V CF              02  FB 001CD          CALLS   #2, IŃSQUE1
                75   17 A2              05  E1 001D2          BBC     #5, 23(R2), 32$                 ; 4580
                                        04  001D7          RET                                     ; 4583
                        50      04  AC  D0 001D8 25$:     MOVL    EVENT, R0                       ; 4594
                        03      30  A0  91 001DC          CMPB    48(R0), #3
                                        01  12 001E0          BNEQ    26$
                                        04  001E2          RET
                     0C AE              03  D0 001E3 26$:     MOVL    #3, OP                          ; 4596
                                        01  DD 001E7          PUSHL   #1
                             10 AE      9F  001E9          PUSHAB  OP
                                        58  DD 001EC          PUSHL   EVENT_ADDRESS
               00000000G 00             03  FB 001EE          CALLS   #3, DBG$WRITE_MEM
                        01              50  E9 001F5          BLBC    R0, 27$
                                        04  001F8          RET
                     000284B4   8F      DD  001F9 27$:     PUSHL   #165044                         ; 4599
               00000000G 00             01  FB 001FF          CALLS   #1, LIB$SIGNAL
                                        04  00206          RET                                     ; 4598
                        51              6B  D0 00207 28$:     MOVL    DBG$RUNFRAME+64, R1             ; 4611
                        50              61  9A 0020A          MOVZBL  (R1), OPCODE
                        52      18      A6  9E 0020D          MOVAB   24(R6), R2                      ; 4614
             03         62              03  E0 00211          BBS     #3, (R2), 30$
                                  00BD  31  00215 29$:     BRW     39$
             7C         62              0A  E0 00218 30$:     BBS     #10, (R2), 37$                  ; 4615
             F1 00000000'   EF          50  E1 0021C          BBC     OPCODE, DBG$OPCODES_STEP_OVER, 29$
             70         62              0A  E0 00224          BBS     #10, (R2), 37$                  ; 4621
                00000FA 8F              50  D1 00228          CMPL    OPCODE, #250                    ; 4624
                                        09  13 0022F          BEQL    31$
                00000FB 8F              50  D1 00231          CMPL    OPCODE, #251
                                        1E  12 00238          BNEQ    33$
                     01 A2              04  88 0023A 31$:     BISB2   #4, 1(R2)                       ; 4636
                           24   A6      D4  0023E          CLRL    36(R6)                          ; 4637
                     03 A2      02  A9  90 00241          MOVB    2(R9), 3(R2)                    ; 4639
                     01 A9    0400  8F  B0 00246          MOVW    #1024, 1(R9)                    ; 4638
                     04 AB              10  88 0024C 32$:     BISB2   #16, DBG$RUNFRAME+68            ; 4641
               00000000G 00             04  88 00250          BISB2   #4, DBG$GV_CONTROL+1            ; 4642
                                        04  00257          RET                                     ; 4626
                                        7E  7C 00258 33$:     CLRQ    -(SP)                           ; 4651
                                        51  DD 0025A          PUSHL   R1                              ; 4652
               00000000G 00             03  FB 0025C          CALLS   #3, DBG$INS_DECODE
                        58              50  D0 00263          MOVL    R0, EVENT_ADDRESS
                40000000 8F             58  D1 00266          CMPL    EVENT_ADDRESS, #1073741824      ; 4657
                                        04  1E 0026D          BGEQU   34$
                                        58  D5 0026F          TSTL    EVENT_ADDRESS                   ; 4658
                                        1B  12 00271          BNEQ    36$
                                        62  95 00273 34$:     TSTB    (R2)                            ; 4671
                                        04  19 00275          BLSS    35$
                                        58  D5 00277          TSTL    EVENT_ADDRESS                   ; 4672
                                        5E  12 00279          BNEQ    40$
                                        6B  DD 0027B 35$:     PUSHL   DBG$RUNFRAME+64                 ; 4674
                                        01  DD 0027D          PUSHL   #1
```

```
                        00028063   8F  DD 0027F          PUSHL   #163939                                    ;
00000000G   00                     03  FB 00285          CALLS   #3, LIB$SIGNAL                           ;
                                   4B  11 0028C          BRB     40$                                      ; 4679
      2C    A6                     58  D0 0028E 36$:     MOVL    EVENT_ADDRESS, 44(R6)                    ; 4689
      01    A2                     04  88 00292          BISB2   #4, 1(R2)                                ; 4694
                                   04  11 00296          BRB     38$                                      ; 4621
            58          2C   A6    D0 00298 37$:     MOVL    44(R6), EVENT_ADDRESS                    ; 4703
      30    A6                     68  90 0029C 38$:     MOVB    (EVENT_ADDRESS), 48(R6)                  ; 4715
            03          30   A6    91 002A0          CMPB    48(R6), #3                               ; 4722
                                   37  13 002A4          BEQL    41$                                      ;
      10    AE                     03  D0 002A6          MOVL    #3, OP                                   ; 4724
                                   01  DD 002AA          PUSHL   #1                                       ;
                          14   AE   9F  002AC          PUSHAB  OP                                       ;
                                   58  DD 002AF          PUSHL   EVENT_ADDRESS                            ;
00000000G   00                     03  FB 002B1          CALLS   #3, DBG$WRITE_MEM                        ;
            22                     50  E8 002B8          BLBS    R0, 41$                                  ;
                                   6B  DD 002BB          PUSHL   DBG$RUNFRAME+64                          ; 4727
                                   01  DD 002BD          PUSHL   #1                                       ;
                        00028063   8F  DD 002BF          PUSHL   #163939                                  ;
00000000G   00                     03  FB 002C5          CALLS   #3, LIB$SIGNAL                           ;
      04    AB                     10  88 002CC          BISB2   #16, DBG$RUNFRAME+68                     ; 4728
      01    A2                     04  8A 002D0          BICB2   #4, 1(R2)                                ; 4729
                                   04  002D4          RET                                              ; 4726
      01    A2                     04  8A 002D5 39$:     BICB2   #4, 1(R2)                                ; 4739
      04    AB                     10  88 002D9 40$:     BISB2   #16, DBG$RUNFRAME+68                     ; 4740
                                   04  002DD 41$:     RET                                              ; 4748

; Routine Size:  734 bytes,    Routine Base:  DBG$CODE + 1A4C
```

```
;  4630          4749  1 GLOBAL ROUTINE DBG$DEACTIVATE_EVENTS: NOVALUE =
;  4631          4750  1 !
;  4632          4751  1 ! FUNCTION
;  4633          4752  1 !       This routine deactivates all set events.
;  4634          4753  1 !
;  4635          4754  1 ! INPUTS
;  4636          4755  1 !       NONE
;  4637          4756  1 !
;  4638          4757  1 ! OUTPUTS
;  4639          4758  1 !       The events are deactivated, including the resetting of breakpoints, the
;  4640          4759  1 !       resetting of protections, etc.
;  4641          4760  1 !
;  4642          4761  1
;  4643          4762  2     BEGIN
;  4644          4763  2
;  4645          4764  2     LOCAL
;  4646          4765  2         EVENT_ENTRY :   REF EVENT$EVENT_DESCRIPTOR, ! Event entry pointer
;  4647          4766  2         PAGE_ENTRY  :   REF EVENT$PAGE_DESCRIPTOR;  ! Page entry pointer
;  4648          4767  2
;  4649          4768  2
;  4650          4769  2     ! As long as were not skipping ACCVIO's, for each entry in the page
;  4651          4770  2     ! queue, un-write protect the page described.
;  4652          4771  2     !
;  4653          4772  2     IF NOT .SKIP_ACCVIOS
;  4654          4773  2     THEN
;  4655          4774  3         BEGIN
;  4656          4775  3         PAGE_ENTRY = .EVENT$PAGE_QUEUE [L_QUEUE_FLINK];
;  4657          4776  3         WHILE .PAGE_ENTRY NEQA EVENT$PAGE_QUEUE DO
;  4658          4777  4             BEGIN
;  4659          4778  4             LOCAL
;  4660          4779  4                 ADDRESS :       VECTOR [2];     ! Address range for protect
;  4661          4780  4
;  4662          4781  4             ADDRESS [0] = .PAGE_ENTRY [EVENT$L_PAGE_ADDRESS];
;  4663          4782  4             ADDRESS [1] = .PAGE_ENTRY [EVENT$L_PAGE_ADDRESS];
;  4664        P 4783  4             IF NOT $SETPRT (INADR       = ADDRESS,
;  4665        P 4784  4                             PROT        = .PAGE_ENTRY [EVENT$B_PAGE_PROTECTION]
;  4666          4785  5                            )
;  4667          4786  4             THEN
;  4668          4787  4                 SIGNAL (DBG$_NOWPROT)
;  4669          4788  4             ELSE
;  4670          4789  4                 PAGE_ENTRY = .PAGE_ENTRY [EVENT$L_PAGE_FLINK];
;  4671          4790  3             END;
;  4672          4791  2         END;
;  4673          4792  2
;  4674          4793  2
;  4675          4794  2     ! Deactivate the all events not 'deleted'.
;  4676          4795  2     !
;  4677          4796  2     EVENT_ENTRY = .EVENT$CMD_QUEUE [L_QUEUE_FLINK];
;  4678          4797  2     WHILE .EVENT_ENTRY NEQA EVENT$CMD_QUEUE [A_QUEUE_ENTRY] DO
;  4679          4798  2         BEGIN
;  4680          4799  2         IF NOT .EVENT_ENTRY [EVENT$V_DELETED]
;  4681          4800  2         THEN
;  4682          4801  2             DBG$DEACTIVATE_EVENT (.EVENT_ENTRY);
;  4683          4802  2         EVENT_ENTRY = .EVENT_ENTRY [EVENT$L_CMD_FLINK];
;  4684          4803  2         END;
;  4685          4804  1     END;
```

```
                                 000C 00000          .ENTRY  DBG$DEACTIVATE_EVENTS, Save R2,R3                ; 4749
                53 00000000'  EF   9E 00002          MOVAB   EVENT$PAGE_QUEUE, R3
                5E            08   C2 00009          SUBL2   #8, SP
                3D 00000000'  EF   E8 0000C          BLBS    SKIP_ACCVIOS, 3$                                ; 4772
                52            63   D0 00013          MOVL    EVENT$PAGE_QUEUE, PAGE_ENTRY                    ; 4775
                50            63   9E 00016 1$:      MOVAB   EVENT$PAGE_QUEUE, R0                            ; 4776
                50            52   D1 00019          CMPL    PAGE_ENTRY, R0
                              32   13 0001C          BEQL    3$
                6E        08  A2   D0 0001E          MOVL    8(PAGE_ENTRY), ADDRESS                          ; 4781
          04    AE        08  A2   D0 00022          MOVL    8(PAGE_ENTRY), ADDRESS+4                        ; 4782
                            7E   D4 00027          CLRL    -(SP)                                           ; 4785
                7E        0E  A2   9A 00029          MOVZBL  14(PAGE_ENTRY), -(SP)
                            7E   7C 0002D          CLRQ    -(SP)
                          10  AE   9F 0002F          PUSHAB  ADDRESS
00000000G       00            05   FB 00032          CALLS   #5, SYS$SETPRT
                0F            50   E8 00039          BLBS    R0, 2$
                    000284C4  8F   DD 0003C          PUSHL   #165060                                         ; 4787
00000000G       00            01   FB 00042          CALLS   #1, LIB$SIGNAL
                              CB   11 00049          BRB     1$
                52            62   D0 0004B 2$:      MOVL    (PAGE_ENTRY), PAGE_ENTRY                        ; 4789
                              C6   11 0004E          BRB     1$                                              ; 4776
                52        D0  A3   D0 00050 3$:      MOVL    EVENT$CMD_QUEUE, EVENT_ENTRY                    ; 4796
                50        D0  A3   9E 00054 4$:      MOVAB   EVENT$CMD_QUEUE, R0                             ; 4797
                50            52   D1 00058          CMPL    EVENT_ENTRY, R0
                              11   13 0005B          BEQL    6$
                          17  A2   95 0005D          TSTB    23(EVENT_ENTRY)                                 ; 4799
                              07   19 00060          BLSS    5$
                              52   DD 00062          PUSHL   EVENT_ENTRY                                     ; 4801
                0000V     CF        01   FB 00064          CALLS   #1, DBG$DEACTIVATE_EVENT
                52                  62   D0 00069 5$:      MOVL    (EVENT_ENTRY), EVENT_ENTRY                  ; 4802
                                    E6   11 0006C          BRB     4$                                          ; 4797
                                         04 0006E 6$:      RET                                                 ; 4804

; Routine Size:  111 bytes,    Routine Base:  DBG$CODE + 1D2A
```

```
;4687    4805  1 GLOBAL ROUTINE DBG$DEACTIVATE_EVENT(EVENT_ENTRY): NOVALUE =
;4688    4806  1 !
;4689    4807  1 ! FUNCTION
;4690    4808  1 !       This routine deactivates an event.
;4691    4809  1 !
;4692    4810  1 ! INPUTS
;4693    4811  1 !       EVENT_ENTRY :           Pointer to an event entry.
;4694    4812  1 !
;4695    4813  1 ! OUTPUTS
;4696    4814  1 !       The event is deactivated, including the resetting of breakpoints, the
;4697    4815  1 !       resetting of protections, etc.
;4698    4816  1 !
;4699    4817  1
;4700    4818  2     BEGIN
;4701    4819  2
;4702    4820  2     MAP
;4703    4821  2         EVENT_ENTRY :   REF EVENT$EVENT_DESCRIPTOR; ! Event entry pointer
;4704    4822  2
;4705    4823  2     LOCAL
;4706    4824  2         EVENT_ADDRESS : REF VECTOR [, BYTE],        ! Event byte address
;4707    4825  2         USERS_FP :      REF BLOCK [, BYTE],         ! User's current FP
;4708    4826  2         STATUS;                                     ! Status value
;4709    4827  2
;4710    4828  2
;4711    4829  2     ! If this is an IGNOR entry, ignore it !
;4712    4830  2     !
;4713    4831  2     IF .EVENT_ENTRY [EVENT$B_CMD_TYPE] EQLU EVENT$K_TYPE_IGNOR
;4714    4832  2     THEN
;4715    4833  2         RETURN;
;4716    4834  2
;4717    4835  2
;4718    4836  2     ! Case on the kind of event entry, to determine whether how to activate
;4719    4837  2     ! the entry.
;4720    4838  2     !
;4721    4839  2     CASE .EVENT_ENTRY [EVENT$B_CMD_KIND]
;4722    4840  2     FROM EVENT$K_KIND_ACC TO EVENT$K_KIND_EXC OF
;4723    4841  2         SET
;4724    4842  2         [EVENT$K_KIND_ACC]:
;4725    4843  2
;4726    4844  2             ! Case on the access sub-kind....
;4727    4845  2             !
;4728    4846  2             CASE .EVENT_ENTRY [EVENT$B_SUB_KIND]
;4729    4847  2             FROM EVENT$K_ACC_READ TO EVENT$K_ACC_RTRN OF
;4730    4848  2                 SET
;4731    4849  2
;4732    4850  2
;4733    4851  2                 [EVENT$K_ACC_READ]:
;4734    4852  2                     ;
;4735    4853  2
;4736    4854  2
;4737    4855  2                 [EVENT$K_ACC_WRIT]:
;4738    4856  2                     ;
;4739    4857  2
;4740    4858  2
;4741    4859  2                 [EVENT$K_ACC_MDFY]:
;4742    4860  2                     ;
;4743    4861  2
```

```
;  4744          4862  2
;  4745          4863  2
;  4746          4864  2                    ! Execution Access means that we've got to replace a BPT
;  4747          4865  2                    ! instruction at the address indicated.
;  4748          4866  2                    !
;  4749          4867  2                    [EVENT$K_ACC_EXEC]:
;  4750          4868  3                        BEGIN
;  4751          4869  3
;  4752          4870  3
;  4753          4871  3                        ! Get the event's address.
;  4754          4872  3                        !
;  4755          4873  3                        EVENT_ADDRESS = .EVENT_ENTRY [EVENT$L_ADDRESS];
;  4756          4874  3
;  4757          4875  3
;  4758          4876  3                        ! Finally, overwrite the BPT instruction with the original
;  4759          4877  3                        ! opcode into that address.  Note that we will NOT do this
;  4760          4878  3                        ! if the orignal is a BPT.
;  4761          4879  3                        !
;  4762          4880  3                        IF .EVENT_ENTRY [EVENT$B_OPCODE] NEQU BPT_INSTRUCTION
;  4763          4881  3                        THEN
;  4764        P 4882  3                            IF NOT WRITE_OPCODE (.EVENT_ADDRESS,
;  4765        P 4883  3                                                 .EVENT_ENTRY [EVENT$B_OPCODE]
;  4766          4884  4                                                )
;  4767          4885  3                            THEN
;  4768          4886  4                                BEGIN
;  4769          4887  4                                SIGNAL (DBG$_NOWOPCO);
;  4770          4888  4                                RETURN;
;  4771          4889  3                                END;
;  4772          4890  2                        END;
;  4773          4891  2
;  4774          4892  2
;  4775          4893  2                    ! Return Access means that we've got to replace a BPT
;  4776          4894  2                    ! instruction at the address indicated - or, if this
;  4777          4895  2                    ! is was a STEP command, turn of the dirty bit in the
;  4778          4896  2                    ! user's PSW.
;  4779          4897  2                    !
;  4780          4898  2                    [EVENT$K_ACC_RTRN]:
;  4781          4899  3                        BEGIN
;  4782          4900  3                        IF .EVENT_ENTRY [EVENT$B_CMD_TYPE] EQLU
;  4783          4901  3                            EVENT$K_TYPE_SSINT
;  4784          4902  3                        THEN
;  4785          4903  4                            BEGIN
;  4786          4904  4                            LOCAL BIT_15_CNT: REF VECTOR[,BYTE];
;  4787          4905  4                            BIT_15_CNT = .EVENT_ENTRY [EVENT$L_FP_RET_CNT];
;  4788          4906  4                            IF .BIT_15_CNT[0] GEQ 1 THEN DBG$GB_SET_WATCH_FLAG = TRUE;
;  4789          4907  3                            END;
;  4790          4908  3
;  4791          4909  3
;  4792          4910  3                        ! Depending on whether this was a STEP command or not...
;  4793          4911  3                        !
;  4794          4912  4                        IF (.EVENT_ENTRY [EVENT$B_CMD_TYPE] EQLU
;  4795          4913  4                            EVENT$K_TYPE_STEPS
;  4796          4914  3                           ) OR
;  4797          4915  4                           (.EVENT_ENTRY [EVENT$B_CMD_TYPE] EQLU
;  4798          4916  4                            EVENT$K_TYPE_SKIPS
;  4799          4917  3                           ) OR
;  4800          4918  4                           (.EVENT_ENTRY [EVENT$B_CMD_TYPE] EQLU
```

```
;  4801          4919  4                    EVENT$K_TYPE_SSINT
;  4802          4920  4                    )
;  4803          4921  3                THEN
;  4804          4922  4                    BEGIN
;  4805          4923  4
;  4806          4924  4
;  4807          4925  4                    ! Turn off bit 15 in the user's
;  4808          4926  4                    ! runframe so we don't get another
;  4809          4927  4                    ! ROPRAND fault on the return.
;  4810          4928  4                    !
;  4811          4929  4                    USERS_FP = .EVENT_ENTRY [EVENT$L_USERS_FP];
;  4812          4930  4                    IF .USERS_FP NEQ 0
;  4813          4931  4                    THEN
;  4814          4932  5                        BEGIN
;  4815          4933  5                        MATCH_RIGHT_CALL_FRAME(.EVENT_ENTRY);
;  4816          4934  5                        USERS_FP = .EVENT_ENTRY [EVENT$L_USERS_FP];
;  4817          4935  5                        (USERS_FP [SF$W_SAVE_PSW])<15,1,0> = FALSE;
;  4818          4936  4                        END;
;  4819          4937  4                    END
;  4820          4938  3                ELSE
;  4821          4939  4                    BEGIN
;  4822          4940  4
;  4823          4941  4
;  4824          4942  4                    ! Get the event's address.
;  4825          4943  4                    !
;  4826          4944  4                    EVENT_ADDRESS = .EVENT_ENTRY [EVENT$L_ADDRESS];
;  4827          4945  4
;  4828          4946  4
;  4829          4947  4                    ! Finally, overwrite the BPT instruction with the
;  4830          4948  4                    ! original opcode into that address.  Note that we
;  4831          4949  4                    ! will NOT do this if the orignal is a BPT.
;  4832          4950  4                    !
;  4833          4951  4                    IF .EVENT_ENTRY [EVENT$B_OPCODE] NEQU BPT_INSTRUCTION
;  4834          4952  4                    THEN
;  4835        P 4953  4                        IF NOT WRITE_OPCODE (.EVENT_ADDRESS,
;  4836        P 4954  4                                            .EVENT_ENTRY [EVENT$B_OPCODE]
;  4837          4955  5                                           )
;  4838          4956  4                        THEN
;  4839          4957  5                            BEGIN
;  4840          4958  5                            SIGNAL (DBG$_NOWOPCO);
;  4841          4959  5                            RETURN;
;  4842          4960  4                            END;
;  4843          4961  3                    END;
;  4844          4962  2                END;
;  4845          4963  2            TES;
;  4846          4964  2
;  4847          4965  2
;  4848          4966  2        [EVENT$K_KIND_INS]:
;  4849          4967  2            BEGIN
;  4850          4968  2
;  4851          4969  2
;  4852          4970  2            ! If this step was stepping over (had a BPT set), remove the
;  4853          4971  2            ! BPT instruction.
;  4854          4972  2            !
;  4855          4973  2            IF .EVENT_ENTRY [EVENT$V_STEP_OVER] AND
;  4856          4974  2               .EVENT_ENTRY [EVENT$V_STEP_BPT]
;  4857          4975  3            THEN
```

```
;  4858          4976  4                    BEGIN
;  4859          4977  4
;  4860          4978  4
;  4861          4979  4                    ! Get the event's address.
;  4862          4980  4                    !
;  4863          4981  4                    EVENT_ADDRESS = .EVENT_ENTRY [EVENT$L_ADDRESS];
;  4864          4982  4
;  4865          4983  4
;  4866          4984  4                    ! Finally, overwrite the BPT instruction with the original
;  4867          4985  4                    ! opcode into that address.  Note that we will NOT do this
;  4868          4986  4                    ! if the orignal is a BPT.
;  4869          4987  4                    !
;  4870          4988  4                    IF .EVENT_ENTRY [EVENT$B_OPCODE] NEQU BPT_INSTRUCTION
;  4871          4989  4                    THEN
;  4872        P 4990  4                        IF NOT WRITE_OPCODE (.EVENT_ADDRESS,
;  4873        P 4991  4                                             .EVENT_ENTRY [EVENT$B_OPCODE]
;  4874          4992  5                                            )
;  4875          4993  4                        THEN
;  4876          4994  5                            BEGIN
;  4877          4995  5                            SIGNAL (DBG$_NOWOPCO);
;  4878          4996  5                            RETURN;
;  4879          4997  4                            END;
;  4880          4998  3                    END;
;  4881          4999  2                END;
;  4882          5000  2
;  4883          5001  2
;  4884          5002  2            [EVENT$K_KIND_EXC]:
;  4885          5003  2                ;
;  4886          5004  2            TES;
;  4887          5005  1        END;


                                        003C 00000          .ENTRY  DBG$DEACTIVATE_EVENT, Save R2,R3,R4,R5      ; 4805
                                  55 00000000G   00  9E 00002       MOVAB   DBG$WRITE_MEM, R5                   ;
                                  5E             0C  C2 00009       SUBL2   #12, SP                             ;
                                  52       04    AC  D0 0000C       MOVL    EVENT_ENTRY, R2                     ; 4831
                                  05       14    A2  91 00010       CMPB    20(R2), #5                          ;
                                                7E  13 00014       BEQL    10$                                 ;
                            02          00    15  A2  8F 00016       CASEB   21(R2), #0, #2                      ; 4839
                          00BA        0087        0006     0001B 1$:  .WORD   2$-1$,-                             ;
                                                                             11$-1$,-                           ;
                                                                             14$-1$                             ;
                            04          00    16  A2  8F 00021 2$:   CASEB   22(R2), #0, #4                      ; 4847
                  0008      00AF        00AF      00AF     00026 3$:  .WORD   14$-3$,-                            ;
                                                  0027     0002E              14$-3$,-                           ;
                                                                              14$-3$,-                          ;
                                                                              4$-3$,-                           ;
                                                                              6$-3$                             ;
                                                        04 00030       RET                                      ;
                                  54       2C    A2  D0 00031 4$:   MOVL    44(R2), EVENT_ADDRESS               ; 4873
                                  03       30    A2  91 00035       CMPB    48(R2), #3                          ; 4880
                                                79  13 00039       BEQL    12$                                 ;
                                  6E       30    A2  9A 0003B       MOVZBL  48(R2), OP                          ; 4884
                                                01  DD 0003F       PUSHL   #1                                  ;
```

```
                         04  AE  9F 00041        PUSHAB  OP                              ;
                             54  DD 00044 5$:    PUSHL   EVENT_ADDRESS                   ;
                    65       03  FB 00046        CALLS   #3, DBG$WRITE_MEM               ;
                    7C       50  E9 00049        BLBC    R0, 13$                         ;
                                 04 0004C        RET                                     ; 4887
                             51  D4 0004D 6$:    CLRL    R1                              ; 4900
                    06   14  A2  91 0004F        CMPB    20(R2), #6                      ;
                             11  12 00053        BNEQ    7$                              ;
                             51  D6 00055        INCL    R1                              ;
                    50   2C  A2  D0 00057        MOVL    44(R2), BIT_15_CNT              ; 4905
                             60  95 0005B        TSTB    (BIT_15_CNT)                    ; 4906
                             07  13 0005D        BEQL    7$                              ;
          00000000'  EF      01  90 0005F        MOVB    #1, DBG$GB_SET_WATCH_FLAG       ;
                    03   14  A2  91 00066 7$:    CMPB    20(R2), #3                      ; 4912
                             09  13 0006A        BEQL    8$                              ;
                    04   14  A2  91 0006C        CMPB    20(R2), #4                      ; 4915
                             03  13 00070        BEQL    8$                              ;
                    17       51  E9 00072        BLBC    R1, 9$                          ; 4918
                    53   24  A2  D0 00075 8$:    MOVL    36(R2), USERS_FP                ; 4929
                             5A  13 00079        BEQL    14$                             ; 4930
                             52  DD 0007B        PUSHL   R2                              ; 4933
             0000V  CF       01  FB 0007D        CALLS   #1, MATCH_RIGHT_CALL_FRAME      ;
                    53   24  A2  D0 00082        MOVL    36(R2), USERS_FP                ; 4934
                05  A3   80  8F  8A 00086        BICB2   #128, 5(USERS_FP)               ; 4935
                                 04 0008B        RET                                     ; 4912
                    54   2C  A2  D0 0008C 9$:    MOVL    44(R2), EVENT_ADDRESS           ; 4944
                    03   30  A2  91 00090        CMPB    48(R2), #3                      ; 4951
                             3F  13 00094 10$:   BEQL    14$                             ;
                04  AE   30  A2  9A 00096        MOVZBL  48(R2), OP                      ; 4955
                             01  DD 0009B        PUSHL   #1                              ;
                         08  AE  9F 0009D        PUSHAB  OP                              ;
                             A2  11 000A0        BRB     5$                              ;
          2E        18  A2   03  E1 000A2 11$:   BBC     #3, 24(R2), 14$                 ; 4973
          29        19  A2   02  E1 000A7        BBC     #2, 25(R2), 14$                 ; 4974
                    54   2C  A2  D0 000AC        MOVL    44(R2), EVENT_ADDRESS           ; 4981
                    03   30  A2  91 000B0        CMPB    48(R2), #3                      ; 4988
                             1F  13 000B4 12$:   BEQL    14$                             ;
                08  AE   30  A2  9A 000B6        MOVZBL  48(R2), OP                      ; 4992
                             01  DD 000BB        PUSHL   #1                              ;
                         0C  AE  9F 000BD        PUSHAB  OP                              ;
                             54  DD 000C0        PUSHL   EVENT_ADDRESS                   ;
                    65       03  FB 000C2        CALLS   #3, DBG$WRITE_MEM               ;
                    0D       50  E8 000C5        BLBS    R0, 14$                         ;
                 000284BC    8F  DD 000C8 13$:   PUSHL   #165052                         ; 4995
          00000000G  00      01  FB 000CE        CALLS   #1, LIB$SIGNAL                  ;
                                 04 000D5 14$:   RET                                     ; 5005

; Routine Size:  214 bytes,    Routine Base:  DBG$CODE + 1D99
```

```
;  4889          5006  1 GLOBAL ROUTINE DBG$EXCEPTION_HANDLER(SIGNAL_ARG_PTR, MECHAN_ARG_PTR) =
;  4890          5007  1 !
;  4891          5008  1 !  FUNCTIONAL DESCRIPTION:
;  4892          5009  1 !
;  4893          5010  1 !       Exception analyzer called by the primary vector exception handler,
;  4894          5011  1 !       which is a MARS routine found in DBGSTART.MAR.  The MARS routine
;  4895          5012  1 !       immediately resignals if the user program was not running.  Otherwise
;  4896          5013  1 !       it saves the registers of the user program and disables ASTs for the
;  4897          5014  1 !       time that DEBUG is running.
;  4898          5015  1 !
;  4899          5016  1 !       Then it calls this routine, where the exception is analyzed for the
;  4900          5017  1 !       type of exception.  Breakpoint, trace, and ACCVIO traps are given
;  4901          5018  1 !       special handling, which usually ends with control being passed to the
;  4902          5019  1 !       user.  If the breakpoint, trace, or ACCVIO trap was illegal, then the
;  4903          5020  1 !       exception is resignaled unless the user has asked for control on every
;  4904          5021  1 !       exception.
;  4905          5022  1 !
;  4906          5023  1 !       Some trace traps cause an interim halt that requires some action, but
;  4907          5024  1 !       doesn't pass control back to the user.  After checking the validity of
;  4908          5025  1 !       these trace traps, the value SS$_CONTINUE is returned.
;  4909          5026  1 !
;  4910          5027  1 !       After the exception is analyzed and it is determined that immediate
;  4911          5028  1 !       resignaling or continuing is not desired, the user_proc routine is
;  4912          5029  1 !       called.  This routine accepts user commands either from prespecified
;  4913          5030  1 !       action commands from breakpoints, or interactively from the terminal.
;  4914          5031  1 !       Eventually, a command is given that either causes the user program to
;  4915          5032  1 !       continue or DEBUG to exit. If the user program is to continue, the
;  4916          5033  1 !       value returned from user_proc is SS$_CONTINUE, and that value is
;  4917          5034  1 !       passed back to the MARS handler.
;  4918          5035  1 !
;  4919          5036  1 !       If an exception occurs during DEBUG processing, the exception handler
;  4920          5037  1 !       is FINAL_HANDL, not this routine.
;  4921          5038  1 !
;  4922          5039  1 !
;  4923          5040  1 !  FORMAL PARAMETERS:
;  4924          5041  1 !
;  4925          5042  1 !       SIGNAL_ARG_PTR  - address of block that contains at least four
;  4926          5043  1 !                         longwords.  The pertinent words are the exception
;  4927          5044  1 !                         name, the PC at the time of the exception, and the PSL
;  4928          5045  1 !                         at the time of the exception.  The name is always the
;  4929          5046  1 !                         second longword, the PC and the PSL the next to last
;  4930          5047  1 !                         and last respectively.
;  4931          5048  1 !
;  4932          5049  1 !       MECHAN_ARG_PTR  - address of block that contains five longwords.  The
;  4933          5050  1 !                         pertinent words are the saved R0 and R1.  They are in
;  4934          5051  1 !                         the fourth and fifth longwords respectively.  Neither
;  4935          5052  1 !                         is used at this time.
;  4936          5053  1 !
;  4937          5054  1 !
;  4938          5055  1 !  IMPLICIT INPUTS:
;  4939          5056  1 !
;  4940          5057  1 !       Some variable number of additional arguments may exist between the
;  4941          5058  1 !       exception name and the PC.  Flags indicating the validity of TBITs and
;  4942          5059  1 !       breakpoints are referenced.  The flag DBG$GB_RESIGNAL causes illegal
;  4943          5060  1 !       exceptions to be resignaled if the flag is set to true.
;  4944          5061  1 !
;  4945          5062  1 !
```

```
;  4946          5063  1 ! IMPLICIT OUTPUTS:
;  4947          5064  1 !
;  4948          5065  1 !       The TBIT in the RUNFRAME PSL may be changed.
;  4949          5066  1 !
;  4950          5067  1 !
;  4951          5068  1 ! ROUTINE VALUE:
;  4952          5069  1 !
;  4953          5070  1 !       SS$_RESIGNAL or SS$_CONTINUE for resignaling and continuing
;  4954          5071  1 !       respectively.
;  4955          5072  1 !
;  4956          5073  1 !
;  4957          5074  1 ! SIDE EFFECTS:
;  4958          5075  1 !
;  4959          5076  1 !       Any number of things.
;  4960          5077  1 !
;  4961          5078  2     BEGIN
;  4962          5079  2     MAP
;  4963          5080  2         MECHAN_ARG_PTR : REF BLOCK[,BYTE],
;  4964          5081  2         SIGNAL_ARG_PTR : REF BLOCK[,BYTE];
;  4965          5082  2     BUILTIN PROBER;
;  4966          5083  2     LABEL scan;
;  4967          5084  2     LOCAL
;  4968          5085  2         EVENT :          REF EVENT$EVENT_DESCRIPTOR, ! Event entry pointer
;  4969          5086  2         LAST1 :          REF EVENT$EVENT_DESCRIPTOR, ! Event entry pointer
;  4970          5087  2         SKIPS_ENTRY :    REF EVENT$EVENT_DESCRIPTOR, ! Skip entry pointer
;  4971          5088  2         ACTIVE_EVENT,                                ! Event match flag
;  4972          5089  2         EXCEPTION_TYPE,
;  4973          5090  2         USERS_PC :       REF BLOCK [, BYTE],
;  4974          5091  2         USERS_FP :       REF BLOCK [, BYTE],
;  4975          5092  2         OPCODE,
;  4976          5093  2         ORIG_TBIT,                                   ! Original TBIT value
;  4977          5094  2         STATUS;
;  4978          5095  2
;  4979          5096  2
;  4980          5097  2     ! Save the original value of the TBIT.
;  4981          5098  2     !
;  4982          5099  2     ORIG_TBIT = .DBG$RUNFRAME[DBG$V_TBIT];
;  4983          5100  2
;  4984          5101  2
;  4985          5102  2     ! Save the exception type (name);
;  4986          5103  2     !
;  4987          5104  2     EXCEPTION_TYPE = .SIGNAL_ARG_PTR [CHF$L_SIG_NAME];
;  4988          5105  2
;  4989          5106  2
;  4990          5107  2     ! Save the User's PC.
;  4991          5108  2     !
;  4992          5109  2     USERS_PC = .DBG$RUNFRAME [DBG$L_USER_PC];
;  4993          5110  2     IF NOT PROBER(%REF(0),%REF(1),.USERS_PC) THEN OPCODE = 0
;  4994          5111  4     ELSE IF (((OPCODE = .(.USERS_PC)<0,8,0>) GEQU %X'FD')
;  4995          5112  3                 AND PROBER(%REF(0),%REF(2),.USERS_PC))
;  4996          5113  2     THEN OPCODE = %X'100' + .(.USERS_PC)<8,8,0>;
;  4997          5114  2
;  4998          5115  2
;  4999          5116  2     ! If we got here via a TBIT, see if we're at one of the special DEBUG
;  5000          5117  2     ! locations.  If so, just return SS$_CONTINUE.
;  5001          5118  2     !
;  5002          5119  2     IF .EXCEPTION_TYPE EQLU SS$_TBIT
```

```
;  5003     5120  2      THEN
;  5004     5121  2          BEGIN
;  5005     5122  2          IF .USERS_PC EQLA DBG$PSEUDO_EXIT OR
;  5006     5123  2             .USERS_PC EQLA DBG$TERM_HANDLR OR
;  5007     5124  2             .USERS_PC EQLA DBG$USER_EXIT   OR
;  5008     5125  2             .USERS_PC EQLA PRIM_HANDL_2    OR
;  5009     5126  2             .USERS_PC EQLA DBG$PSEUDO_SSI
;  5010     5127  2          THEN
;  5011     5128  3              BEGIN
;  5012     5129  3              DBG$RUNFRAME [DBG$V_TBIT] = FALSE;
;  5013     5130  3              DBG$GL_OUTPRAB [RAB$V_CCO] = TRUE;
;  5014     5131  3              RETURN SS$_CONTINUE;
;  5015     5132  3              END;
;  5016     5133  2
;  5017     5134  2          END;
;  5018     5135  2
;  5019     5136  2
;  5020     5137  2      ! If we got here via an Opcode Request....
;  5021     5138  2      !
;  5022     5139  2      IF .EXCEPTION_TYPE EQLU SS$_DBGOPCREQ
;  5023     5140  2      THEN
;  5024     5141  2          BEGIN
;  5025     5142  2
;  5026     5143  2
;  5027     5144  2          ! This signal is generated by the library routine LIB$GET_OPCODE.
;  5028     5145  2          ! It is used to inquire what the real instruction was before the
;  5029     5146  2          ! debugger replaced the instruction with a BPT.
;  5030     5147  2          !
;  5031     5148  2          ! (The emulator needs this capability).
;  5032     5149  2          !
;  5033     5150  2          ! Note that the opcode involved was restored when we deactivated
;  5034     5151  2          ! the events.  This will have to be redone when we stop
;  5035     5152  2          ! automatically deactivating....
;  5036     5153  2          !
;  5037     5154  2          DBG$DEACTIVATE_EVENTS();
;  5038     5155  2          (.SIGNAL_ARG_PTR [12,0,32,0]) <0, 8, 0> = .(.SIGNAL_ARG_PTR [8,0,32,0]) <0, 8, 0>;
;  5039     5156  2          DBG$ACTIVATE_EVENTS();
;  5040     5157  2          RETURN SS$_CONTINUE;
;  5041     5158  2          END;
;  5042     5159  2
;  5043     5160  2
;  5044     5161  2      ! *****SSI
;  5045     5162  2      ! If we got here via System service interception...
;  5046     5163  2      ! Build a SSI enty on the Event List, if this system service calls from
;  5047     5164  2      ! user program, and nobody else has turned on bit 15 in saved PSW in
;  5048     5165  2      ! System Service FP.  (This means that TDBG has seen this already, bit
;  5049     5166  2      ! 15 is set, now is SDBG's turn to examine this singal).  Record the PC
;  5050     5167  2      ! to report watch At, unprotected the pages...
;  5051     5168  2      !
;  5052     5169  2      IF .EXCEPTION_TYPE EQLU DBG$_SS_INT
;  5053     5170  2      THEN
;  5054     5171  2          BEGIN
;  5055     5172  2          LOCAL
;  5056     5173  2              ADDRESS : VECTOR [2],                   ! Address range for protect
;  5057     5174  2              BIT_15_CNT: REF VECTOR[,BYTE],          ! Has bit 15 turned on?
;  5058     5175  2              BIT_15_CNT_SSV: REF VECTOR[,BYTE],      ! Are we called from System
;  5059     5176  2                                                      !   Service's system service
```

```
;  5060          5177  ;                                          !  call?
;  5061          5178  ;           PAGE_ENTRY: REF EVENT$PAGE_DESCRIPTOR,! Page entry pointer
;  5062          5179  ;           SIGARG: REF VECTOR[,LONG],      ! Info. comes in with DBG$_SS_INT
;  5063          5180  ;           SSINT_ENTRY: REF EVENT$EVENT_DESCRIPTOR,! SSI event entry
;  5064          5181  ;           USERS_FP: REF BLOCK[,BYTE];     ! System service call frame
;  5065          5182  ;
;  5066          5183  ;
;  5067          5184  ;       ! Get arg. list.
;  5068          5185  ;       !
;  5069          5186  ;       SIGARG = SIGNAL_ARG_PTR[CHF$L_SIG_ARG1];
;  5070          5187  ;
;  5071          5188  ;       ! Caused from System service's system call, simply ignore it.
;  5072          5189  ;       ! note: this variable is a communication variable, it is seen
;  5073          5190  ;       ! by DBG, TDBG and SDBG.
;  5074          5191  ;       !
;  5075          5192  ;       BIT_15_CNT_SSV = .SIGARG[5];
;  5076          5193  ;       IF .BIT_15_CNT_SSV[0] GTR 1 THEN RETURN SS$_CONTINUE;
;  5077          5194  ;
;  5078          5195  ;       ! This type of entry is already built for somebody else (caused from
;  5079          5196  ;       ! the same source, went through different level).  Ignore it.
;  5080          5197  ;       ! note: this varialbe is a communication variable, it is seen
;  5081          5198  ;       ! by DBG, TDBG and SDBG.
;  5082          5199  ;       !
;  5083          5200  ;       BIT_15_CNT = .SIGARG[4];
;  5084          5201  ;       IF .BIT_15_CNT[0] GEQ 1 THEN RETURN SS$_CONTINUE;
;  5085          5202  ;
;  5086          5203  ;       ! We'll take it.  Mark the cycle begins.  Build event entry, place
;  5087          5204  ;       ! it at the head of the event list.
;  5088          5205  ;       !
;  5089          5206  ;       DBG$GB_SET_WATCH_FLAG = TRUE;
;  5090          5207  ;       USERS_FP = .SIGARG[3];
;  5091          5208  ;       EVENT = .EVENT$CMD_QUEUE [L_QUEUE_FLINK];
;  5092          5209  ;       SSINT_ENTRY = DBG$GET_MEMORY(EVENT$K_EVENT_DESCRIPTOR_SIZE);
;  5093          5210  ;       INSQUE (.SSINT_ENTRY, .EVENT[EVENT$L_CMD_BLINK]);
;  5094          5211  ;       SSINT_ENTRY [EVENT$B_CMD_TYPE] = EVENT$K_TYPE_SSINT;
;  5095          5212  ;       SSINT_ENTRY [EVENT$B_CMD_KIND] = EVENT$K_KIND_ACC;
;  5096          5213  ;       SSINT_ENTRY [EVENT$B_SUB_KIND] = EVENT$K_ACC_RTRN;
;  5097          5214  ;       SSINT_ENTRY [EVENT$V_ONCE_ONLY] = TRUE;
;  5098          5215  ;       SSINT_ENTRY [EVENT$L_EXC_FLINK] = .SSINT_ENTRY;
;  5099          5216  ;       SSINT_ENTRY [EVENT$L_EXC_BLINK] = .SSINT_ENTRY;
;  5100          5217  ;       SSINT_ENTRY [EVENT$L_USERS_FP] = .USERS_FP;
;  5101          5218  ;       (USERS_FP [SF$W_SAVE_PSW]) <15,1,0> = TRUE;
;  5102          5219  ;       SSINT_ENTRY [EVENT$L_SSV_COUNT] = .BIT_15_CNT_SSV;
;  5103          5220  ;       SSINT_ENTRY [EVENT$L_FP_RET_CNT] = .BIT_15_CNT;
;  5104          5221  ;       SSINT_ENTRY [EVENT$L_USERS_PC] = .USERS_FP[SF$L_SAVE_PC];
;  5105          5222  ;       WATCH_PC = .USERS_FP[SF$L_SAVE_PC];
;  5106          5223  ;       BIT_15_CNT[0] = .BIT_15_CNT[0] + 1;
;  5107          5224  ;       IF .BIT_15_CNT[0] EQL 1
;  5108          5225  ;       THEN
;  5109          5226  4;           BEGIN
;  5110          5227  4;           PAGE_ENTRY = .EVENT$PAGE_QUEUE [L_QUEUE_FLINK];
;  5111          5228  4;           WHILE .PAGE_ENTRY NEQA EVENT$PAGE_QUEUE DO
;  5112          5229  5;               BEGIN
;  5113          5230  5;               ADDRESS [0] = .PAGE_ENTRY [EVENT$L_PAGE_ADDRESS];
;  5114          5231  5;               ADDRESS [1] = .PAGE_ENTRY [EVENT$L_PAGE_ADDRESS];
;  5115        P 5232  5;               IF NOT $SETPRT (INADR   = ADDRESS,
;  5116        P 5233  5;                               PROT    = .PAGE_ENTRY [EVENT$B_PAGE_PROTECTION]
```

```
;  5117          5234  6                          )
;  5118          5235  5                  THEN
;  5119          5236  5                      SIGNAL (DBG$_NOWPROT)
;  5120          5237  5                  ELSE
;  5121          5238  5                      PAGE_ENTRY = .PAGE_ENTRY [EVENT$L_PAGE_FLINK];
;  5122          5239  4                  END;
;  5123          5240  3              END;
;  5124          5241  3
;  5125          5242  3          RETURN SS$_CONTINUE;
;  5126          5243  3          END;
;  5127          5244  2
;  5128          5245  2
;  5129          5246  2      ! If this is an exception break or trace, we'll flag it later.
;  5130          5247  2      !
;  5131          5248  2      DBG$GB_EXC_BRE_FLAG = FALSE;
;  5132          5249  2
;  5133          5250  2
;  5134          5251  2      ! Clear the event-match flag, to indicate that we have as yet found no
;  5135          5252  2      ! matching event entry for this exception.
;  5136          5253  2      !
;  5137          5254  2      ACTIVE_EVENT = FALSE;
;  5138          5255  2
;  5139          5256  2
;  5140          5257  2      ! Set the type-source flag, so that we will display the source line at
;  5141          5258  2      ! the first opportunity.
;  5142          5259  2      !
;  5143          5260  2      TYPE_SOURCE = TRUE;
;  5144          5261  2
;  5145          5262  2
;  5146          5263  2      ! Clear the STOP_USER flag, set when a break is valid, or step is over.
;  5147          5264  2      ! This flag is usually set in PROCESS_EVENT when a break breaks, or a
;  5148          5265  2      ! step stops....
;  5149          5266  2      !
;  5150          5267  2      EVENT$B_STOP_USER = FALSE;
;  5151          5268  2
;  5152          5269  2
;  5153          5270  2      ! Set the flag allowing user commands, i.e. no GO, STEP, EXIT, or
;  5154          5271  2      ! CONTINUE commands have been executed yet.  We have to do it here
;  5155          5272  2      ! in case there are no DO's evaluated, since we test it later....
;  5156          5273  2      !
;  5157          5274  2      DBG$GB_TAKE_CMD = TRUE;
;  5158          5275  2
;  5159          5276  2
;  5160          5277  2      ! Deactivate all events.  This will most probably be changed to take
;  5161          5278  2      ! advantage of not HAVING to turn them all off, but later, LATER !
;  5162          5279  2      !
;  5163          5280  2      DBG$DEACTIVATE_EVENTS ();
;  5164          5281  2      IF .EXCEPTION_TYPE EQLU SS$_DEBUG
;  5165          5282  2      THEN
;  5166          5283  3          BEGIN
;  5167          5284  3
;  5168          5285  3
;  5169          5286  3          ! This is the phony way into the user control. This exception name
;  5170          5287  3          ! is passed by the DEBUG startup routine, and by the final handler
;  5171          5288  3          ! when an exception generated in a user program has not been dealt
;  5172          5289  3          ! with by any user handlers (and so has been reported by the DEBUG
;  5173          5290  3          ! final handler),  when the user is passed control in order to fix
```

```
;  5174     5291  3          ! up the condition that caused the problem.
;  5175     5292  3          !
;  5176     5293  3          ! We'll act as though a BREAKpoint was encountered.
;  5177     5294  3          !
;  5178     5295  3          EVENT$B_STOP_USER = TRUE;
;  5179     5296  3          ACTIVE_EVENT = TRUE;
;  5180     5297  3          END;
;  5181     5298  3
;  5182     5299  3
;  5183     5300  3      ! If the user program has completed, cancel any pending STEP events.
;  5184     5301  3      !
;  5185     5302  3      IF .DBG$RUNFRAME[DBG$L_USER_PC] EQL 0
;  5186     5303  3      THEN
;  5187     5304  3          BEGIN
;  5188     5305  3          EVENT = EVENT$CMD_QUEUE[A_QUEUE_ENTRY];
;  5189     5306  3          WHILE TRUE DO
;  5190     5307  4              BEGIN
;  5191     5308  4              EVENT = .EVENT[EVENT$L_CMD_FLINK];
;  5192     5309  4              IF .EVENT EQLA EVENT$CMD_QUEUE[A_QUEUE_ENTRY] THEN EXITLOOP;
;  5193     5310  5              IF (.EVENT[EVENT$B_CMD_TYPE] EQL EVENT$K_TYPE_STEPS)
;  5194     5311  4              THEN
;  5195     5312  4                  EVENT[EVENT$V_DELETED] = TRUE;
;  5196     5313  4
;  5197     5314  4              END;
;  5198     5315  3
;  5199     5316  3          END
;  5200     5317  3
;  5201     5318  3      ELSE
;  5202     5319  3          BEGIN
;  5203     5320  3
;  5204     5321  3
;  5205     5322  3      ! Based on exception name, handle the exception....
;  5206     5323  3      !
;  5207     5324  3      SELECTONE .EXCEPTION_TYPE OF
;  5208     5325  3          SET
;  5209     5326  3
;  5210     5327  3          [SS$_ROPRAND,SS$_DEBUG,SS$_BREAK,SS$_TBIT]:
;  5211     5328  3            IF (.EXCEPTION_TYPE NEQ SS$_DEBUG) OR
;  5212     5329  3               .DBG$GV_CONTROL[DBG$V_CONTROL_STOP]
;  5213     5330  3            THEN
;  5214     5331  4              BEGIN
;  5215     5332  4
;  5216     5333  4
;  5217     5334  4              ! A BPT or TBIT gets us here.  These are two of the
;  5218     5335  4              ! exceptions DEBUG uses to make events happen, and so
;  5219     5336  4              ! are treated specially here.
;  5220     5337  4              !
;  5221     5338  4              ! Point to the current last entry on the queue.  We'll only
;  5222     5339  4              ! examine entries until :
;  5223     5340  4              ! a) we're at the actual end of the queue, or
;  5224     5341  4              ! b) we've looked at the last entry we knew about when we started.
;  5225     5342  4              ! This is because processing a DO might add entries that we
;  5226     5343  4              ! then want to ignore....
;  5227     5344  4              !
;  5228     5345  4              LAST1 = .EVENT$CMD_QUEUE [L_QUEUE_BLINK];
;  5229     5346  4
;  5230     5347  4
```

```
;  5231     5343  4                ! For each event entry in the queue, check to see if it is
;  5232     5349  4                ! active....
;  5233     5350  4                !
;  5234     5351  4                EVENT = .EVENT$CMD_QUEUE [L_QUEUE_FLINK];
;  5235     5352  4                WHILE .EVENT NEQA EVENT$CMD_QUEUE [A_QUEUE_ENTRY] DO
;  5236     5353  5                    BEGIN
;  5237     5354  5
;  5238     5355  5
;  5239     5356  5                    ! Make sure that the entry isn't deleted but not removed....
;  5240     5357  5                    !
;  5241     5358  5                    IF NOT .EVENT [EVENT$V_DELETED]
;  5242     5359  5                    THEN
;  5243     5360  6                        BEGIN
;  5244     5361  6
;  5245     5362  6
;  5246     5363  6                        ! If this entry is to be IGNORed....
;  5247     5364  6                        !
;  5248     5365  6                        IF .EVENT [EVENT$B_CMD_TYPE] EQLU EVENT$K_TYPE_IGNOR
;  5249     5366  6                        THEN
;  5250     5367  6
;  5251     5368  6
;  5252     5369  6                            ! IGNORed entries are always deleted.  If this wasn't
;  5253     5370  6                            ! a TBIT exception, the entry will be re-entered when
;  5254     5371  6                            ! the associated parent-event is re-activated ('cause
;  5255     5372  6                            ! all IGNORed entries are used to TBIT over some
;  5256     5373  6                            ! instruction).
;  5257     5374  6                            !
;  5258     5375  7                            BEGIN
;  5259     5376  7                            EVENT [EVENT$V_DELETED] = TRUE;
;  5260     5377  7
;  5261     5378  7
;  5262     5379  7                            ! Ignored entries are active if this is a TBIT.
;  5263     5380  7                            !
;  5264     5381  7                            IF .EXCEPTION_TYPE EQLU SS$_TBIT
;  5265     5382  7                            THEN
;  5266     5383  7                                ACTIVE_EVENT = TRUE;
;  5267     5384  7                            END
;  5268     5385  7
;  5269     5386  6                        ELSE
;  5270     5387  6
;  5271     5388  6                        ! This is a 'real' event, not an IGNORed one.
;  5272     5389  6                        !
;  5273     5390  6                        BEGIN
;  5274     5391  7
;  5275     5392  7
;  5276     5393  7                        ! Case on the kind of event entry, to determine whether
;  5277     5394  7                        ! the exception is applicable, and if the entry is
;  5278     5395  7                        ! active.
;  5279     5396  7                        !
;  5280     5397  7                        CASE .EVENT [EVENT$B_CMD_KIND]
;  5281     5398  7                        FROM EVENT$K_KIND_ACC TO EVENT$K_KIND_EXC OF
;  5282     5399  7                            SET
;  5283     5400  7                            [EVENT$K_KIND_ACC]:
;  5284     5401  7
;  5285     5402  7
;  5286     5403  7                                ! Case on the access sub-kind....
;  5287     5404  7
```

```
;  5288          5405  7
;  5289          5406  7
;  5290          5407  7
;  5291          5408  7
;  5292          5409  7
;  5293          5410  7
;  5294          5411  7
;  5295          5412  7
;  5296          5413  7
;  5297          5414  7
;  5298          5415  7
;  5299          5416  7
;  5300          5417  7
;  5301          5418  7
;  5302          5419  7
;  5303          5420  7
;  5304          5421  7
;  5305          5422  7
;  5306          5423  7
;  5307          5424  7
;  5308          5425  7
;  5309          5426  7
;  5310          5427  7
;  5311          5428  7
;  5312          5429  7
;  5313          5430  7
;  5314          5431  7
;  5315          5432  7
;  5316          5433  7
;  5317          5434  8
;  5318          5435  8
;  5319          5436  8
;  5320          5437  8
;  5321          5438  8
;  5322          5439  8
;  5323          5440  8
;  5324          5441  8
;  5325          5442  8
;  5326          5443  8
;  5327          5444  8
;  5328          5445  8
;  5329          5446  8
;  5330          5447  8
;  5331          5448  8
;  5332          5449  8
;  5333          5450  8
;  5334          5451  8
;  5335          5452  8
;  5336          5453  8
;  5337          5454  8
;  5338          5455  8
;  5339          5456  8
;  5340          5457  8
;  5341          5458  8
;  5342          5459  8
;  5343          5460  8
;  5344          5461  8

!
CASE .EVENT [EVENT$B_SUB_KIND]
FROM EVENT$K_ACC_READ TO EVENT$K_ACC_RTRN OF
    SET

    [EVENT$K_ACC_READ]:
        ;

    [EVENT$K_ACC_WRIT]:
        ;

    [EVENT$K_ACC_MDFY]:

        ! If we're skipping ACCVIOS (meaning
        ! we had one, and it's page address
        ! was one we're watching for....
        ! *****SSI
        ! If we're returned from system
        ! service, or during system service
        ! cycle...
        !
        IF .SKIP_ACCVIOS OR
           .SKIP_WATCHES OR
           .DBG$GB_SET_WATCH_FLAG
        THEN
            BEGIN
            LOCAL
                LENGTH;

            ! We've got an active event.
            !
            ACTIVE_EVENT = TRUE;

            ! Get the byte length of the
            ! entry's value.  Note that we
            ! actually get its bit length and
            ! round up.  This should be okay
            ! 'cause the values saved in a
            ! value descriptor are padded out
            ! to fill bytes.
            !
            LENGTH =
                DBG$DATA_LENGTH
                    (EVENT [EVENT$A_VMSDESC]);
            LENGTH = (.LENGTH + 7) ^ -3;
            GET_WATCH_POINT_VALUE(.EVENT);
            OLDVALUE =
                .EVENT [EVENT$L_VALDESC];

            ! Now, compare the two values, and
```

```
8                          ! process the event is they're the
8                          ! same.
8                          !
8                          IF CH$EQL
8                              (.LENGTH,
8                               OLDVALUE [DBG$A_VALUE_ADDRESS],
8                               .LENGTH,
8                               NEWVALUE [DBG$A_VALUE_ADDRESS],
8                               0
8                              )
8                          THEN
8                              DBG$NFREE_DESC(.NEWVALUE)
8                          ELSE
9                              BEGIN
9
9
9                              ! Process the entry.
9                              !
9                              IF .SKIP_WATCHES OR
9                                 .DBG$GB_SET_WATCH_FLAG
9                              THEN
9                                  ACCVIO_PC = .WATCH_PC;
9                              STATUS = DBG$PROCESS_EVENT
9                                         (.EVENT,
9                                          .SIGNAL_ARG_PTR
9                                         );
9
9                              ! Update the value descriptor in the event entry to refl
9                              ! new value.
9                              ! Then, since we are finished with the value descriptor
9                              ! free up the memory.
9                              !
9                              CH$MOVE
9                                  (.LENGTH,
9                                   NEWVALUE [DBG$A_VALUE_ADDRESS],
9                                   OLDVALUE [DBG$A_VALUE_ADDRESS]
9                                  );
9                              DBG$NFREE_DESC(.NEWVALUE);
9
9                              ! If we have been pre-
9                              ! empted by a GO, STEP,
9                              ! CONTINUE, or EXIT, we will
9                              ! stop processing.
9                              !
9                              IF .STATUS
9                              THEN
9                                  EXITLOOP;
8                              END;
7                          END;
7
7                  ! Process an /EXECUTION (BREAK/TRACE)
7                  ! entry.
7                  !
7                  [EVENT$K_ACC_EXEC]:
8                      BEGIN
8
```

```
;  5402          5519  8
;  5403          5520  8
;  5404          5521  8
;  5405          5522  8
;  5406          5523  8
;  5407          5524  8
;  5408          5525  8
;  5409          5526  9
;  5410          5527  9
;  5411          5528  9
;  5412          5529  9
;  5413          5530  9
;  5414          5531  9
;  5415          5532  9
;  5416          5533  9
;  5417          5534  9
;  5418          5535  9
;  5419          5536  9
;  5420          5537  9
;  5421          5538  9
;  5422          5539  9
;  5423          5540  9
;  5424          5541  9
;  5425          5542  9
;  5426          5543  8
;  5427          5544  8
;  5428          5545  7
;  5429          5546  7
;  5430          5547  7
;  5431          5548  7
;  5432          5549  7
;  5433          5550  7
;  5434          5551  7
;  5435          5552  8
;  5436          5553  8
;  5437          5554  8
;  5438          5555  8
;  5439          5556  8
;  5440          5557  8
;  5441          5558  8
;  5442          5559  8
;  5443          5560  8
;  5444          5561  8
;  5445          5562  8
;  5446          5563  8
;  5447          5564  8
;  5448          5565  9
;  5449          5566  9
;  5450          5567  9
;  5451          5568  9
;  5452          5569  9
;  5453          5570  9
;  5454          5571  9
;  5455          5572  9
;  5456          5573  9
;  5457          5574  9
;  5458          5575  9

        ! Match the address of the eventpoint
        ! with the current PC for an active
        ! event...
        !
        IF .EVENT [EVENT$L_ADDRESS] EQLA .USERS_PC
        THEN
            BEGIN

            ! The event is active, so process
            ! the entry.  If on return we have
            ! been pre-empted by a GO, STEP,
            ! CONTINUE, or EXIT, we will stop
            ! processing.
            !
            ACTIVE_EVENT = TRUE;
            IF DBG$PROCESS_EVENT
                    (.EVENT,
                     .SIGNAL_ARG_PTR
                    )
            THEN
                EXITLOOP;

            END;

        END;

    ! Process a /RETURN (BREAK/TRACE/STEPS/
    ! SKIPS) entry.
    !
    [EVENT$K_ACC_RTRN]:
        BEGIN

        ! Based on the event command type....
        !
        SELECTONE .EVENT [EVENT$B_CMD_TYPE] OF
            SET

            ! [BREAK!TRACE]/RETURN
            !
            [EVENT$K_TYPE_BREAK,
             EVENT$K_TYPE_TRACE]:
                BEGIN

                ! Match the address of the
                ! eventpoint with the current
                ! PC for an active event...
                ! Also make sure the exception is not ROPRAND
                ! (this is in case the user sets a
                ! return break at the RET).
                !
                IF (.EVENT [EVENT$L_ADDRESS] EQLA .USERS_PC) AND
```

```
;  5459          5576 10                        (.EXCEPTION_TYPE NEQ SS$_ROPRAND)
;  5460          5577  9                    THEN
;  5461          5578 10                        BEGIN
;  5462          5579 10
;  5463          5580 10                        ! The event is active.
;  5464          5581 10                        !
;  5465          5582 10                        ACTIVE_EVENT = TRUE;
;  5466          5583 10
;  5467          5584 10                        ! Get a new event descriptor
;  5468          5585 10                        ! and link it into the
;  5469          5586 10                        ! command queue. Note that
;  5470          5587 10                        ! this entry is placed in
;  5471          5588 10                        ! the queue just in front
;  5472          5589 10                        ! of its parent, and should
;  5473          5590 10                        ! be encountered first !!!
;  5474          5591 10                        !
;  5475          5592 10                        SKIPS_ENTRY =
;  5476          5593 10                            DBG$GET_MEMORY
;  5477          5594 10                                (EVENT$K_EVENT_DESCRIPTOR_SIZE);
;  5478          5595 10                        INSQUE (.SKIPS_ENTRY,
;  5479          5596 10                                .EVENT [EVENT$L_CMD_BLINK]
;  5480          5597 10                               );
;  5481          5598 10
;  5482          5599 10                        ! Setup this entry to be a
;  5483          5600 10                        ! /RETURN skip-event,
;  5484          5601 10                        ! linking to the event
;  5485          5602 10                        ! entry causing this mess.
;  5486          5603 10                        !
;  5487          5604 10                        SKIPS_ENTRY [EVENT$B_CMD_TYPE] =
;  5488          5605 10                            EVENT$K_TYPE_SKIPS;
;  5489          5606 10                        SKIPS_ENTRY [EVENT$B_CMD_KIND] =
;  5490          5607 10                            EVENT$K_KIND_ACC;
;  5491          5608 10                        SKIPS_ENTRY [EVENT$B_SUB_KIND] =
;  5492          5609 10                            EVENT$K_ACC_RTRN;
;  5493          5610 10                        SKIPS_ENTRY [EVENT$V_ONCE_ONLY] =
;  5494          5611 10                            TRUE;
;  5495          5612 10                        INSQUEI (.SKIPS_ENTRY, .EVENT);
;  5496          5613 10
;  5497          5614 10                        ! Initialize the FP to 0 so it gets
;  5498          5615 10                        ! setup when activated.
;  5499          5616 10                        !
;  5500          5617 10                        SKIPS_ENTRY [EVENT$L_USERS_FP] = 0;
;  5501          5618 10                        END;
;  5502          5619 10                    END;
;  5503          5620 10
;  5504          5621 10                ! [BREAK!TRACE/RETURN skips and
;  5505          5622  9                ! STEP/RETURN
;  5506          5623  8                !
;  5507          5624  8                [EVENT$K_TYPE_SKIPS,
;  5508          5625  8                 EVENT$K_TYPE_STEPS]:
;  5509          5626  8                    BEGIN
;  5510          5627  8                    SELECTONE .EXCEPTION_TYPE OF
;  5511          5628  8
;  5512          5629  8
;  5513          5630  8
;  5514          5631  9
;  5515          5632  9
```

```
;  5516          5633  9                        SET
;  5517          5634  9                        [SS$_ROPRAND]:
;  5518          5635 10                            BEGIN
;  5519          5636 10                            LOCAL
;  5520          5637 10                                RIGHT_FP_FLAG;
;  5521          5638 10
;  5522          5639 10                            IF .OPCODE EQL %X'04'
;  5523          5640 10                            THEN
;  5524          5641 10                                RIGHT_FP_FLAG =  MATCH_RIGHT_CALL_FRAME(.EVE
;  5525          5642 10                            ELSE
;  5526          5643 10                                RIGHT_FP_FLAG = FALSE;
;  5527          5644 10
;  5528          5645 10                            IF .RIGHT_FP_FLAG
;  5529          5646 10                            THEN
;  5530          5647 11                                BEGIN
;  5531          5648 11                                ACTIVE_EVENT = TRUE;
;  5532          5649 11
;  5533          5650 11                                ! We've come to the RET instruction that we
;  5534          5651 11                                ! in the process of stepping over a routine.
;  5535          5652 11                                ! want to change our RETURN event back into
;  5536          5653 11                                ! ordinary STEP event, to resume stepping no
;  5537          5654 11                                !
;  5538          5655 11                                IF .EVENT [EVENT$V_STEP_BPT]
;  5539          5656 11                                THEN
;  5540          5657 12                                    BEGIN
;  5541          5658 12                                    EVENT [EVENT$V_STEP_BPT] = FALSE;
;  5542          5659 12                                    EVENT [EVENT$B_CMD_KIND] = EVENT$K_KIND
;  5543          5660 12                                    EVENT [EVENT$B_SUB_KIND] = .EVENT [EVENT
;  5544          5661 12                                    END
;  5545          5662 12
;  5546          5663 12                                ! Process the entry.  If on return we have b
;  5547          5664 12                                ! preempted by a GO,  STEP,  CONTINUE,  or E
;  5548          5665 12                                ! we will stop processing.
;  5549          5666 12                                !
;  5550          5667 11                                ELSE IF DBG$PROCESS_EVENT
;  5551          5668 11                                        (.EVENT,
;  5552          5669 11                                         .SIGNAL_ARG_PTR)
;  5553          5670 11                                THEN
;  5554          5671 11                                    EXITLOOP;
;  5555          5672 11
;  5556          5673 10                                END;
;  5557          5674 10
;  5558          5675  9                            END;
;  5559          5676  9
;  5560          5677  9                        [SS$_TBIT]:
;  5561          5678  9                            IF .EVENT [EVENT$V_STEP_BPT]
;  5562          5679  9                               THEN ACTIVE_EVENT = TRUE;
;  5563          5680  9
;  5564          5681  9                        [OTHERWISE]:
;  5565          5682  9                            0;
;  5566          5683  9                        TES;
;  5567          5684  8                    END;
;  5568          5685  8
;  5569          5686  8                ! *****SSI
;  5570          5687  8                ! We are examing the SSI entry was built
;  5571          5688  8                ! at DBG$_SS_INT signal...
;  5572          5689  8                !
```

```
;  5573          5690  8                        [EVENT$K_TYPE_SSINT]:
;  5574          5691  9                            BEGIN
;  5575          5692  9
;  5576          5693  9                            ! Well?  Caused from Reserved Operand
;  5577          5694  9                            ! fault?
;  5578          5695  9                            !
;  5579          5696 10                            IF (.DBG$RUNFRAME [DBG$L_USER_FP] EQLA
;  5580          5697 10                                .EVENT [EVENT$L_USER$_FP])
;  5581          5698 10                            AND (.OPCODE EQLU %X'04')
;  5582          5699  9                            THEN
;  5583          5700 10                                BEGIN
;  5584          5701 10                                EVENT [EVENT$V_DELETED] = TRUE;
;  5585          5702 10                                ACTIVE_EVENT = TRUE;
;  5586          5703 10                                EVENT$B_STOP_USER = FALSE;
;  5587          5704 10                                DBG$GB_TAKE_CMD = FALSE;
;  5588          5705 10                                EXITLOOP;
;  5589          5706  9                                END;
;  5590          5707  8                            END;
;  5591          5708  8                        TES;
;  5592          5709  7                    END;
;  5593          5710  7                TES;
;  5594          5711  7
;  5595          5712  7
;  5596          5713  7            [EVENT$K_KIND_INS]:
;  5597          5714  8                BEGIN
;  5598          5715  8
;  5599          5716  8                ! If this is a TBIT (or DEBUG) or BREAK
;  5600          5717  8                ! exception when stepping over, process
;  5601          5718  8                ! a step.
;  5602          5719  8                !
;  5603          5720  8                IF (SELECTONE .EXCEPTION_TYPE OF
;  5604          5721  9                       SET
;  5605          5722  9                       [SS$_TBIT, SS$_DEBUG]:
;  5606          5723  9                           IF .EVENT [EVENT$V_STEP_BPT]
;  5607          5724  9                           THEN
;  5608          5725  9                               IF .EVENT [EVENT$L_ADDRESS] EQLA .USERS_PC
;  5609          5726  9                               THEN
;  5610          5727  9                                   TRUE
;  5611          5728  9                               ELSE
;  5612          5729  9                                   FALSE
;  5613          5730  9                           ELSE
;  5614          5731  9                               TRUE;
;  5615          5732  9                       [SS$_BREAK]:
;  5616          5733  9                           IF .EVENT [EVENT$V_STEP_BPT]
;  5617          5734  9                           THEN
;  5618          5735  9                               IF .EVENT [EVENT$L_ADDRESS] EQLA .USERS_PC
;  5619          5736  9                               THEN
;  5620          5737  9                                   TRUE
;  5621          5738  9                               ELSE
;  5622          5739  9                                   FALSE
;  5623          5740  9                           ELSE
;  5624          5741  9                               FALSE;
;  5625          5742  9                       [OTHERWISE]:
;  5626          5743  9                           FALSE;
;  5627          5744  9                       TES
;  5628          5745  9                   )
;  5629          5746  9
```

```
;  5630          5747  8  THEN
;  5631          5748  9      BEGIN
;  5632          5749  9
;  5633          5750  9      ! The event is active, regardless.
;  5634          5751  9      !
;  5635          5752  9      ACTIVE_EVENT = TRUE;
;  5636          5753  9
;  5637          5754  9      ! If this was a BPT exception, turn off the
;  5638          5755  9      ! STEP_BPT flag - we're done stepping over.
;  5639          5756  9      !
;  5640          5757  9      IF .EXCEPTION_TYPE EQLU SS$_BREAK
;  5641          5758  9      THEN
;  5642          5759  9          EVENT [EVENT$V_STEP_BPT] = FALSE;
;  5643          5760  9
;  5644          5761  9      ! Depending on the step type (line/instruction) process the step.
;  5645          5762  9      ! We also check here that either we are stepping /system, or else we
;  5646          5763  9      ! not be in system space. That is, if we find that we are stepping n
;  5647          5764  9      ! and that we are indeed in system space, then the IF condition will
;  5648          5765  9      ! false and we will not process the event. We actually use ADDRESS G
;  5649          5766  9      ! as our determination of what is system space, since most system se
;  5650          5767  9      ! are actually entered through P1 space now.
;  5651          5768  9      !
;  5652          5769  9      IF ((NOT .EVENT [EVENT$V_STEP_NOSYSTEM]) OR
;  5653          5770  9          (.DBG$RUNFRAME[DBG$L_USER_PC] LSS P1_SPACE))
;  5654          5771 10                  AND
;  5655          5772 10         (SELECTONE .EVENT [EVENT$B_SUB_KIND] OF
;  5656          5773  9              SET
;  5657          5774 10             [EVENT$K_INS_EVRY]:
;  5658          5775 10                 TRUE;
;  5659          5776 10
;  5660          5777 10             [EVENT$K_INS_CALL,EVENT$K_INS_BRAN,EVENT$K_INS_USER]:
;  5661          5778 10                 BEGIN
;  5662          5779 10                 BIND
;  5663          5780 11                     OP_MAP = EVENT[EVENT$L_OPCODE_LIST] : REF BITVECTOR
;  5664          5781 11
;  5665          5782 11                 ! Check the opcode bitmap to see if the current
;  5666          5783 11                 ! instruction is one we're looking for.
;  5667          5784 11                 !
;  5668          5785 11                 .OP_MAP [.OPCODE]
;  5669          5786 11                 END;
;  5670          5787 11
;  5671          5788 11             [EVENT$K_INS_LINE]:
;  5672          5789 10
;  5673          5790 10                 ! In the following parenthesized expression,
;  5674          5791 10                 ! TRUE =>  process this event, which generally means
;  5675          5792 10                 !          stop tbit-ing and take user commands,
;  5676          5793 10                 ! FALSE => don't take this event,
;  5677          5794 10                 !          which generally means keep tbit-ing.
;  5678          5795 10                 !
;  5679          5796 10                 ! If we are still in the range given by
;  5680          5797 10                 ! LO_PC, HI_PC then don't take the event -
;  5681          5798 10                 ! keep tbit-ing.
;  5682          5799 10                 !
;  5683          5800 10                 (IF (.USERS_PC GEQA .EVENT [EVENT$L_STEP_LO_PC])
;  5684          5801 10
;  5685          5802 10
;  5686          5803 12
```

```
;  5687          5804 11                                                 AND
;  5688          5805 12                                    (.USERS_PC LEQA .EVENT [EVENT$L_STEP_HI_PC])
;  5689          5806 11                                THEN FALSE
;  5690          5807 11                                ELSE
;  5691          5808 12                                    BEGIN
;  5692          5809 12                                    LOCAL
;  5693          5810 12                                        LINENO,
;  5694          5811 12                                        MODRST,
;  5695          5812 12                                        STATUS,
;  5696          5813 12                                        STMTNO;
;  5697          5814 12
;  5698          5815 12                                    ! The PC is out of the LO_PC, HI_PC range.
;  5699          5816 12                                    ! Call the PC lookup routine to get a new range.
;  5700          5817 12                                    ! If the PC lookup routine returns FALSE
;  5701          5818 12                                    ! (0 or 2) then the lookup failed. In this
;  5702          5819 12                                    ! case re-initialize the range to 1,0 so
;  5703          5820 12                                    ! that we immediately are out of the range
;  5704          5821 12                                    ! again after the next tbit.
;  5705          5822 12                                    !
;  5706          5823 12                                    MODRST = 0;
;  5707          5824 12                                    STATUS = DBG$PC_TO_LINE_LOOKUP(
;  5708          5825 12                                            .USERS_PC,
;  5709          5826 12                                            LINENO,
;  5710          5827 12                                            STMTNO,
;  5711          5828 12                                            EVENT[EVENT$L_STEP_LO_PC],
;  5712          5829 12                                            EVENT[EVENT$L_STEP_HI_PC],
;  5713          5830 12                                            MODRST);
;  5714          5831 12                                    IF NOT .STATUS THEN
;  5715          5832 13                                        BEGIN
;  5716          5833 13                                        EVENT[EVENT$L_STEP_LO_PC] = 1;
;  5717          5834 13                                        EVENT[EVENT$L_STEP_HI_PC] = 0;
;  5718          5835 12                                        END;
;  5719          5836 12
;  5720          5837 12                                    ! The lookup routine can return any of:
;  5721          5838 12                                    ! 0 - no line number information for module
;  5722          5839 12                                    ! 1 - exact match to line
;  5723          5840 12                                    ! 2 - have pc/line tables, but no match for this PC
;  5724          5841 12                                    ! 3 - found line, not exact match.
;  5725          5842 12                                    ! We definitely want to stop and take user commands
;  5726          5843 12                                    ! for 0 (examples are stepping into RTL or module
;  5727          5844 12                                    ! that is not set).
;  5728          5845 12                                    ! We also want to stop for 1 (stepped to the beginni
;  5729          5846 12                                    ! of a new line), and 3 (stepped to the middle of
;  5730          5847 12                                    ! a new line).
;  5731          5848 12                                    ! For 2, we keep going - we have stepped to a
;  5732          5849 12                                    ! between-lines location. RPG generates this kind
;  5733          5850 12                                    ! of code, and doesn't want us to stop at
;  5734          5851 12                                    ! addresses without line numbers.
;  5735          5852 12                                    !
;  5736          5853 13                                    (.STATUS NEQ 2)
;  5737          5854 10                                    END);
;  5738          5855 10                                TES
;  5739          5856 10                                )
;  5740          5857  9                            THEN
;  5741          5858  9
;  5742          5859  9                                ! The entry is active.
;  5743          5860  9
```

```
;  5744          5861  9                    !
;  5745          5862 10                    BEGIN
;  5746          5863 10
;  5747          5864 10                    ! Process the entry.  If on return we
;  5748          5865 10                    ! have been preempted by a GO, STEP,
;  5749          5866 10                    ! CONTINUE, or EXIT, we will stop
;  5750          5867 10                    ! processing.
;  5751          5868 10                    !
;  5752          5869 10                    IF DBG$PROCESS_EVENT
;  5753          5870 10                            (.EVENT,
;  5754          5871 10                             .SIGNAL_ARG_PTR
;  5755          5872 10                            )
;  5756          5873 10                    THEN
;  5757          5874 10                        EXITLOOP;
;  5758          5875 10
;  5759          5876  9                    END;
;  5760          5877  9
;  5761          5878  8                END;
;  5762          5879  8
;  5763          5880  7            END;
;  5764          5881  7
;  5765          5882  7
;  5766          5883  7            [EVENT$K_KIND_EXC]:
;  5767          5884  7                0;
;  5768          5885  7
;  5769          5886  7            TES;
;  5770          5887  7
;  5771          5888  7
;  5772          5889  7        ! Release any temporary memory used during the
;  5773          5890  7        ! processing of this event.  Also release all
;  5774          5891  7        ! Temporary RST Entries that have zero reference
;  5775          5892  7        ! counts.
;  5776          5893  7        !
;  5777          5894  7        DBG$REL_TEMPMEM ();
;  5778          5895  7        DBG$RST_TEMP_RELEASE();
;  5779          5896  6        END;
;  5780          5897  6
;  5781          5898  5    END;
;  5782          5899  5
;  5783          5900  5
;  5784          5901  5    ! If this entry was the 'last' one, exit this loop.  Any others
;  5785          5902  5    ! on the list now came from "SET BREAK ..." commands in some
;  5786          5903  5    ! entry's DO command list....
;  5787          5904  5    !
;  5788          5905  5    IF .EVENT EQLA .LAST1 THEN EXITLOOP;
;  5789          5906  5
;  5790          5907  5
;  5791          5908  5    ! Link to the next entry.
;  5792          5909  5    !
;  5793          5910  5    EVENT = .EVENT [EVENT$L_CMD_FLINK];
;  5794          5911  4    END;
;  5795          5912  4
;  5796          5913  4
;  5797          5914  4  ! We're not skipping ACCVIO's any longer.
;  5798          5915  4  !
;  5799          5916  4  SKIP_ACCVIOS = FALSE;
;  5800          5917  4  SKIP_WATCHES = FALSE;
```

```
;  5801          5918  3                END;
;  5802          5919
;  5803          5920
;  5804          5921  3            [SS$_ACCVIO]:
;  5805          5922
;  5806          5923
;  5807          5924  3                ! DEBUG uses ACCVIO's for /MODIFY events.  When an ACCVIO occurs,
;  5808          5925  3                ! we search the page queue for a page address match.  If one is
;  5809          5926  3                ! found, we flag the event as active, flag that we're skipping
;  5810          5927  3                ! the ACCVIO, and we're done.
;  5811          5928  3                !
;  5812          5929  4                BEGIN
;  5813          5930  4                LOCAL
;  5814          5931  4                    PAGE_ENTRY :        REF EVENT$PAGE_DESCRIPTOR,  ! Page entry
;  5815          5932  4                    PAGE_ADDRESS;
;  5816          5933  4
;  5817          5934  4                IF NOT .SKIP_ACCVIOS
;  5818          5935  4                THEN
;  5819          5936  5                    BEGIN
;  5820          5937  5
;  5821          5938  5                    ! Get the page address, and point to the 1st page entry.
;  5822          5939  5                    !
;  5823          5940  5                    PAGE_ADDRESS = .SIGNAL_ARG_PTR [12,0,32,0] AND %X'FFFFFE00';
;  5824          5941  5                    PAGE_ENTRY = .EVENT$PAGE_QUEUE [L_QUEUE_FLINK];
;  5825          5942  5
;  5826          5943  5
;  5827          5944  5                    ! Search the page queue for an entry with this address.
;  5828          5945  5                    !
;  5829          5946  5                    WHILE .PAGE_ENTRY NEQA EVENT$PAGE_QUEUE DO
;  5830          5947  5                        BEGIN
;  5831          5948  6                        IF .PAGE_ENTRY [EVENT$L_PAGE_ADDRESS] EQLA .PAGE_ADDRESS
;  5832          5949  6                        THEN
;  5833          5950  6
;  5834          5951  6                            ! Found the matching entry, so flag that we're skipping
;  5835          5952  6                            ! 'cause of an ACCVIO, and exit the search loop.
;  5836          5953  6                            !
;  5837          5954  6                            BEGIN
;  5838          5955  6                            SKIP_ACCVIOS = TRUE;
;  5839          5956  7                            ACTIVE_EVENT = TRUE;
;  5840          5957  7                            EXITLOOP;
;  5841          5958  7                            END
;  5842          5959  7
;  5843          5960  7                        ELSE
;  5844          5961  7                            PAGE_ENTRY = .PAGE_ENTRY [EVENT$L_PAGE_FLINK];
;  5845          5962  6
;  5846          5963  6                        END;
;  5847          5964  6
;  5848          5965  5                    ! Save the current PC for announcing where we were after we've
;  5849          5966  5                    ! finished executing this instruction.
;  5850          5967  5                    !
;  5851          5968  5                    ACCVIO_PC = .USERS_PC;
;  5852          5969  5                    END;
;  5853          5970  5
;  5854          5971  5                END;
;  5855          5972  4
;  5856          5973  4
;  5857          5974  3            END;
```

```
;  5858          5975  3          TES;
;  5859          5976  3
;  5860          5977  3
;  5861          5978  3      ! If no active events were discovered, look through the queue (as we
;  5862          5979  3      ! remember it, sans any new additions) for an exception break.  If
;  5863          5980  3      ! found process it.
;  5864          5981  3      !
;  5865          5982  3      IF NOT .ACTIVE_EVENT
;  5866          5983  3      THEN
;  5867          5984  3          BEGIN
;  5868          5985  4
;  5869          5986  4
;  5870          5987  4          ! For each event entry in the queue, check to see if it is active....
;  5871          5988  4          !
;  5872          5989  4          EVENT = .EVENT$CMD_QUEUE [L_QUEUE_FLINK];
;  5873          5990  4          WHILE .EVENT NEQA EVENT$CMD_QUEUE [A_QUEUE_ENTRY] DO
;  5874          5991  4              BEGIN
;  5875          5992  5
;  5876          5993  5
;  5877          5994  5              ! Make sure that the entry isn't deleted but not removed....
;  5878          5995  5              !
;  5879          5996  5              IF NOT .EVENT [EVENT$V_DELETED]
;  5880          5997  5              THEN
;  5881          5998  5
;  5882          5999  5
;  5883          6000  5                  ! If this is an exception break....
;  5884          6001  5                  !
;  5885          6002  5                  IF .EVENT [EVENT$B_CMD_KIND] EQLU EVENT$K_KIND_EXC
;  5886          6003  5                  THEN
;  5887          6004  5                      BEGIN
;  5888          6005  6
;  5889          6006  6
;  5890          6007  6                      ! The event is active.
;  5891          6008  6                      !
;  5892          6009  6                      ACTIVE_EVENT = TRUE;
;  5893          6010  6
;  5894          6011  6
;  5895          6012  6                      ! This is an exception break or trace, so flag it.
;  5896          6013  6                      !
;  5897          6014  6                      DBG$GB_EXC_BRE_FLAG = TRUE;
;  5898          6015  6
;  5899          6016  6
;  5900          6017  6                      ! Process the event, and scoot (it's the only one).
;  5901          6018  6                      !
;  5902          6019  6                      DBG$PROCESS_EVENT (.EVENT, .SIGNAL_ARG_PTR);
;  5903          6020  6                      EXITLOOP;
;  5904          6021  6                      END;
;  5905          6022  5
;  5906          6023  5
;  5907          6024  5              ! If this entry was the 'last' one, exit this loop.  Any others
;  5908          6025  5              ! on the list now came from "SET BREAK ..." commands in some
;  5909          6026  5              ! entry's DO command list....
;  5910          6027  5              !
;  5911          6028  5              IF .EVENT EQLA .LAST1 THEN EXITLOOP;
;  5912          6029  5
;  5913          6030  5
;  5914          6031  5
```

```
;  5915     6032  5                  ! Link to the next entry.
;  5916     6033  5                  !
;  5917     6034  5                  EVENT = .EVENT [EVENT$L_CMD_FLINK];
;  5918     6035  4                  END;
;  5919     6036  4
;  5920     6037  3              END;
;  5921     6038  3
;  5922     6039  2          END;
;  5923     6040  2
;  5924     6041  2
;  5925     6042  2      ! If an active event was found, meaning we understood the exception,
;  5926     6043  2      ! return check to see if a break/step ended (and the user didn't go
;  5927     6044  2      ! or step in some DO clause).  If this is the case, go back to the
;  5928     6045  2      ! user for commands, otherwise continue.  If an active event wasn't
;  5929     6046  2      ! found, return SS$_RESIGNAL....
;  5930     6047  2      !
;  5931     6048  2      IF .ACTIVE_EVENT
;  5932     6049  2      THEN
;  5933     6050  3          BEGIN
;  5934     6051  3
;  5935     6052  3
;  5936     6053  3          ! If this was not a trace event, then the EVENT$B_STOP_USER flag will
;  5937     6054  3          ! still be on, indicating that we want to accept DEBUG commands.
;  5938     6055  3          ! Also, if the user has typed ctrl/Y, DEBUG, then the STOP flag
;  5939     6056  3          ! will have been turned on, and in this case we also want to take
;  5940     6057  3          ! user commands.
;  5941     6058  3          !
;  5942     6059  3          IF .EVENT$B_STOP_USER OR
;  5943     6060  3             .DBG$GV_CONTROL[DBG$V_CONTROL_STOP]
;  5944     6061  3          THEN
;  5945     6062  4              BEGIN
;  5946     6063  4              LOCAL
;  5947     6064  4                  START_PC :  REF VECTOR [, WORD];
;  5948     6065  4
;  5949     6066  4
;  5950     6067  4              ! Turn off RMS cancel Control-O bit so that user can toggle
;  5951     6068  4              ! Control-O in the normal fashion.
;  5952     6069  4              !
;  5953     6070  4              DBG$GL_OUTPRAB [RAB$V_CCO] = FALSE;
;  5954     6071  4
;  5955     6072  4
;  5956     6073  4              ! Mark that the user program has executed so that all screen
;  5957     6074  4              ! displays will be updated before we take the next command from
;  5958     6075  4              ! the user terminal.  Note that this is suppressed if we just
;  5959     6076  4              ! returned normally from a CALL command routine.  Hence the CALL
;  5960     6077  4              ! command does not update screen displays.  If we are about to
;  5961     6078  4              ! take commands, we also clear the CALL command status flag.
;  5962     6079  4              !
;  5963     6080  4              IF .DBG$GB_CALL_NORMAL_RET NEQ 2
;  5964     6081  4              THEN
;  5965     6082  4                  DBG$GV_CONTROL[DBG$V_CONTROL_SCREEN] = TRUE;
;  5966     6083  4
;  5967     6084  4              IF .DBG$GB_TAKE_CMD THEN DBG$GB_CALL_NORMAL_RET = 0;
;  5968     6085  4
;  5969     6086  4
;  5970     6087  4              ! As long as no commands in the any processed DO clause caused the
;  5971     6088  4              ! program to continue (or restart), the command taking flag will
```

```
;  5972          6089  4              ! still be turned on, and further commands can be processed from
;  5973          6090  4              ! the input device. Otherwise, the while loop below does not
;  5974          6091  4              ! execute at all.
;  5975          6092  4              !
;  5976          6093  4              WHILE .DBG$GB_TAKE_CMD DO
;  5977          6094  4                  DBG$COMMAND_PROC ();
;  5978          6095  4
;  5979          6096  4              END;
;  5980          6097  4
;  5981          6098  4
;  5982          6099  3          ! Activate events....
;  5983          6100  3          !
;  5984          6101  3          DBG$ACTIVATE_EVENTS ();
;  5985          6102  3
;  5986          6103  3
;  5987          6104  3          ! Return the proper status, depending primarily on whether this was
;  5988          6105  3          ! an EXCEPTION break/trace, or a normal type.
;  5989          6106  3          !
;  5990          6107  3          IF .DBG$GB_EXC_BRE_FLAG
;  5991          6108  3          THEN
;  5992          6109  4              BEGIN
;  5993          6110  4              DBG$RUNFRAME[DBG$V_TBIT] = .ORIG_TBIT;
;  5994          6111  4
;  5995          6112  4
;  5996          6113  4              ! Resignal the exception so that handlers can get it
;  5997          6114  4              ! (unless the user said GO <address> to go somewhere
;  5998          6115  4              ! else).
;  5999          6116  4              !
;  6000          6117  4              IF NOT .DBG$GB_TAKE_CMD
;  6001          6118  4              THEN
;  6002          6119  4                  IF .DBG$GB_GO_ARG_FLAG
;  6003          6120  4                   THEN
;  6004          6121  4                       RETURN SS$_CONTINUE
;  6005          6122  4                   ELSE
;  6006          6123  4                       RETURN SS$_RESIGNAL
;  6007          6124  4              ELSE
;  6008          6125  4                  RETURN SS$_RESIGNAL
;  6009          6126  4              END
;  6010          6127  3          ELSE
;  6011          6128  3
;  6012          6129  3
;  6013          6130  3              ! Return to the user.
;  6014          6131  3              !
;  6015          6132  3              RETURN SS$_CONTINUE;
;  6016          6133  3          END;
;  6017          6134  2
;  6018          6135  2
;  6019          6136  2      ! Didn't find an active event, so resignal the exception....
;  6020          6137  2      !
;  6021          6138  2      DBG$ACTIVATE_EVENTS ();
;  6022          6139  2      DBG$RUNFRAME[DBG$V_TBIT] = .ORIG_TBIT;
;  6023          6140  2      RETURN SS$_RESIGNAL
;  6024          6141  1      END;
```

```
                                                 0FFC 00000          .ENTRY  DBG$EXCEPTION_HANDLER, Save R2,R3,R4,R5,R6,-:  5006
                                                                             R7,R8,R9,R10,R11
                                                   24  C2 00002          SUBL2   #36, SP
      04   AE 00000000G   00                        04  EF 00005          EXTZV   #4, #1, DBG$RUNFRAME+68, ORIG_TBIT          5099
                                                   52       04  AC  D0 0000F      MOVL    SIGNAL_ARG_PTR, R2                  5104
                                                   5A       04  A2  D0 00013      MOVL    4(R2), EXCEPTION_TYPE
                                                   5B 00000000G 00  D0 00017      MOVL    DBG$RUNFRAME+64, USERS_PC           5109
                                 6B                01          00  0C 0001E      PROBER  #0, #1, (USERS_PC)                  5110
                                                               04  12 00022      BNEQ    1$
                                                               6E  D4 00024      CLRL    OPCODE
                                                               1D  11 00026      BRB     2$
                                                   6E                  6B  9A 00028 1$:   MOVZBL  (USERS_PC), OPCODE          5111
                                      000000FD     8F                  6E  D1 0002B      CMPL    OPCODE, #253
                                                                       11  1F 00032      BLSSU   2$
                                 6B                02                  00  0C 00034      PROBER  #0, #2, (USERS_PC)          5112
                                                                       0B  13 00038      BEQL    2$
                                                   6E       01         AB  9A 0003A      MOVZBL  1(USERS_PC), OPCODE         5113
                                                   6E 00000100         8F  C0 0003E      ADDL2   #256, OPCODE
                                      00000464     8F                  5A  D1 00045 2$:   CMPL    EXCEPTION_TYPE, #1124       5119
                                                                       4D  12 0004C      BNEQ    4$
                                                   50 00000000G        00  9E 0004E      MOVAB   DBG$PSEUDO_EXIT, R0         5122
                                                   50                  5B  D1 00055      CMPL    USERS_PC, R0
                                                                       30  13 00058      BEQL    3$
                                                   50 00000000G        00  9E 0005A      MOVAB   DBG$TERM_HANDLR, R0         5123
                                                   50                  5B  D1 00061      CMPL    USERS_PC, R0
                                                                       24  13 00064      BEQL    3$
                                                   50 00000000G        00  9E 00066      MOVAB   DBG$USER_EXIT, R0           5124
                                                   50                  5B  D1 0006D      CMPL    USERS_PC, R0
                                                                       18  13 00070      BEQL    3$
                                                   50 00000000G        00  9E 00072      MOVAB   PRIM_HANDL_2, R0            5125
                                                   50                  5B  D1 00079      CMPL    USERS_PC, R0
                                                                       0C  13 0007C      BEQL    3$
                                                   50 00000000G        00  9E 0007E      MOVAB   DBG$PSEUDO_SSI, R0          5126
                                                   50                  5B  D1 00085      CMPL    USERS_PC, R0
                                                                       11  12 00088      BNEQ    4$
                 00000000G     00                                    10  8A 0008A 3$:   BICB2   #16, DBG$RUNFRAME+68         5129
                 00000000G     00                 80                 8F  88 00091      BISB2   #128, DBG$GL_OUTPRAB+7      5130
                                                                       18  11 00099      BRB     5$                          5131
                 000006A1      8F                                    5A  D1 0009B 4$:   CMPL    EXCEPTION_TYPE, #1697       5139
                                                                       12  12 000A2      BNEQ    6$
                       FE12    CF                                    00  FB 000A4      CALLS   #0, DBG$DEACTIVATE_EVENTS   5154
                         0C    B2                 0B                 B2  90 000A9      MOVB    @8(R2), @12(R2)             5155
                       FA4B    CF                                    00  FB 000AE      CALLS   #0, DBG$ACTIVATE_EVENTS     5156
                                                                 04FA  31 000B3 5$:   BRW     71$                         5157
                 00028793      8F                                    5A  D1 000B6 6$:   CMPL    EXCEPTION_TYPE, #165779     5169
                                                                       03  13 000BD      BEQL    7$
                                                                 00B6  31 000BF      BRW     11$
                                                   50                  08  A2  9E 000C2 7$:   MOVAB   8(R2), SIGARG              5186
                                                   54                  14  A0  D0 000C6      MOVL    20(SIGARG), BIT_15_CNT_SSV  5192
                                                   01                      64  91 000CA      CMPB    (BIT_15_CNT_SSV), #1        5193
                                                                           E4  1A 000CD      BGTRU   5$
                                                   53                  10  A0  D0 000CF      MOVL    16(SIGARG), BIT_15_CNT      5200
                                                                           63  95 000D3      TSTB    (BIT_15_CNT)                5201
                                                                           DC  12 000D5      BNEQ    5$
                 00000000'     EF                                        01  90 000D7      MOVB    #1, DBG$GB_SET_WATCH_FLAG   5206
                                                   52                  0C  A0  D0 000DE      MOVL    12(SIGARG), USERS_FP        5207
```

```
                    56 00000000'   EF   D0 000E2          MOVL    EVENT$CMD_QUEUE, EVENT                          ; 5208
                               10   DD 000E9              PUSHL   #16                                             ; 5209
          00000000G   00           01   FB 000EB          CALLS   #1, DBG$GET_MEMORY                              ;
                04    B6           60   0E 000F2          INSQUE  (SSINT_ENTRY), @4(EVENT)                        ; 5210
                14    A0           06   B0 000F6          MOVW    #6, 20(SSINT_ENTRY)                             ; 5211
                16    A0           04   90 000FA          MOVB    #4, 22(SSINT_ENTRY)                             ; 5213
                17    A0           04   88 000FE          BISB2   #4, 23(SSINT_ENTRY)                             ; 5214
                08    A0           50   D0 00102          MOVL    SSINT_ENTRY, 8(SSINT_ENTRY)                     ; 5215
                0C    A0           50   D0 00106          MOVL    SSINT_ENTRY, 12(SSINT_ENTRY)                    ; 5216
                24    A0           52   D0 0010A          MOVL    USERS_FP, 36(SSINT_ENTRY)                       ; 5217
                05    A2       80  8F   88 0010E          BISB2   #128, 5(USERS_FP)                               ; 5218
                1C    A0           54   D0 00113          MOVL    BIT_15_CNT_SSV, 28(SSINT_ENTRY)                 ; 5219
                2C    A0           53   D0 00117          MOVL    BIT_15_CNT, 44(SSINT_ENTRY)                     ; 5220
                20    A0       10  A2   D0 0011B          MOVL    16(USERS_FP), 32(SSINT_ENTRY)                   ; 5221
          00000000'   EF       10  A2   D0 00120          MOVL    16(USERS_FP), WATCH_PC                          ; 5222
                               63   96 00128              INCB    (BIT_15_CNT)                                    ; 5223
                    01         63   91 0012A              CMPB    (BIT_15_CNT), #1                                ; 5224
                               84   12 0012D              BNEQ    5$
                    52 00000000'   EF   D0 0012F          MOVL    EVENT$PAGE_QUEUE, PAGE_ENTRY                    ; 5227
                    50 00000000'   EF   9E 00136 8$:      MOVAB   EVENT$PAGE_QUEUE, R0                            ; 5228
                    50             52   D1 0013D          CMPL    PAGE_ENTRY, R0
                                   03   12 00140          BNEQ    9$
                                 046B   31 00142          BRW     71$
                1C  AE       08  A2   D0 00145 9$:      MOVL    8(PAGE_ENTRY), ADDRESS                          ; 5230
                20  AE       08  A2   D0 0014A          MOVL    8(PAGE_ENTRY), ADDRESS+4                        ; 5231
                                 7E   D4 0014F          CLRL    -(SP)                                           ; 5234
                    7E       0E  A2   9A 00151          MOVZBL  14(PAGE_ENTRY), -(SP)
                                 7E   7C 00155          CLRQ    -(SP)
                             2C  AE   9F 00157          PUSHAB  ADDRESS
          00000000G   00             05   FB 0015A      CALLS   #5, SYS$SETPRT
                    0F               50   E8 00161      BLBS    R0, 10$
                         00028404    8F   DD 00164      PUSHL   #165060                                         ; 5236
          00000000G   00             01   FB 0016A      CALLS   #1, LIB$SIGNAL                                  ;
                                     C3   11 00171      BRB     8$                                              ;
                    52               62   D0 00173 10$:      MOVL    (PAGE_ENTRY), PAGE_ENTRY                    ; 5238
                                     BE   11 00176      BRB     8$                                              ; 5228
                         00000000G   00   94 00178 11$:      CLRB    DBG$GB_EXC_BRE_FLAG                         ; 5248
                                     59   D4 0017E      CLRL    ACTIVE_EVENT                                    ; 5254
          00000000'   EF             01   D0 00180      MOVL    #1, TYPE_SOURCE                                 ; 5260
                         00000000'   EF   94 00187      CLRB    EVENT$B_STOP_USER                               ; 5267
          00000000G   00             01   90 0018D      MOVB    #1, DBG$GB_TAKE_CMD                             ; 5274
                FD22  CF             00   FB 00194      CALLS   #0, DBG$DEACTIVATE_EVENTS                       ; 5280
          0000046C  8F               5A   D1 00199      CMPL    EXCEPTION_TYPE, #T32                            ; 5281
                                     0A   12 001A0      BNEQ    12$
          00000000'   EF             01   90 001A2      MOVB    #1, EVENT$B_STOP_USER                           ; 5295
                    59               01   D0 001A9      MOVL    #1, ACTIVE_EVENT                                ; 5296
                         00000000G   00   D5 001AC 12$:      TSTL    DBG$RUNFRAME+64                             ; 5302
                                     26   12 001B2      BNEQ    15$
                    56 00000000'   EF   9E 001B4      MOVAB   EVENT$CMD_QUEUE, EVENT                          ; 5305
                    56             66   D0 001BB 13$:      MOVL    (EVENT), EVENT                              ; 5308
                    50 00000000'   EF   9E 001BE      MOVAB   EVENT$CMD_QUEUE, R0                             ; 5309
                    50             56   D1 001C5      CMPL    EVENT, R0
                                   03   12 001C8      BNEQ    14$
                                 0378   31 001CA      BRW     65$
                    03       14  A6   91 001CD 14$:      CMPB    20(EVENT), #3                               ; 5310
                                 E8   12 001D1      BNEQ    13$
                17  A6       80  8F   88 001D3      BISB2   #128, 23(EVENT)                                   ; 5312
```

```
                                                E1  11 001D8        BRB     13$                                  ; 5306
                   00000414  8F                 5A  D1 001DA 15$:   CMPL    EXCEPTION_TYPE, #1044                ; 5327
                                                1E  13 001E1        BEQL    16$                                  ;
                   00000454  8F                 5A  D1 001E3        CMPL    EXCEPTION_TYPE, #1108                ;
                                                15  13 001EA        BEQL    16$                                  ;
                   00000464  8F                 5A  D1 001EC        CMPL    EXCEPTION_TYPE, #1124                ;
                                                0C  13 001F3        BEQL    16$                                  ;
                   0000046C  8F                 5A  D1 001F5        CMPL    EXCEPTION_TYPE, #1132                ;
                                                03  13 001FC        BEQL    16$                                  ;
                                              02B8  31 001FE        BRW     58$                                  ;
                   0000046C  8F                 5A  D1 00201 16$:   CMPL    EXCEPTION_TYPE, #1132                ; 5328
                                                0B  12 00208        BNEQ    17$                                  ;
                03 00000000G 00                 01  E0 0020A        BBS     #1, DBG$GV_CONTROL+1, 17$            ; 5329
                                              02EE  31 00212        BRW     62$                                  ;
                         08  AE 00000000'       EF  D0 00215 17$:   MOVL    EVENT$CMD_QUEUE+4, LAST1             ; 5345
                             56 00000000'       EF  D0 0021D        MOVL    EVENT$CMD_QUEUE, EVENT               ; 5351
                             50 00000000'       EF  9E 00224 18$:   MOVAB   EVENT$CMD_QUEUE, R0                  ; 5352
                             50                 56  D1 0022B        CMPL    EVENT, R0                            ;
                                                03  12 0022E        BNEQ    19$                                  ;
                                              027E  31 00230        BRW     57$                                  ;
                             58          14     A6  9E 00233 19$:   MOVAB   20(EVENT), R8                        ; 5358
                                                68  D5 00237        TSTL    (R8)                                 ;
                                                16  19 00239        BLSS    20$                                  ;
                             05                 68  91 0023B        CMPB    (R8), #5                             ; 5365
                                                14  12 0023E        BNEQ    21$                                  ;
                         03  A8          80     8F  88 00240        BISB2   #128, 3(R8)                          ; 5376
                   00000464  8F                 5A  D1 00245        CMPL    EXCEPTION_TYPE, #1124                ; 5381
                                                03  12 0024C        BNEQ    20$                                  ;
                             59                 01  D0 0024E        MOVL    #1, ACTIVE_EVENT                     ; 5383
                                              0251  31 00251 20$:   BRW     56$                                  ; 5365
             02              00          01     A8  8F 00254 21$:   CASEB   1(R8), #0, #2                        ; 5398
           023E            018A               0006     00259 22$:   .WORD   23$-22$,-                            ;
                                                                            40$-22$,-                            ;
                                                                            55$-22$                              ;
             04              00          02     A8  8F 0025F 23$:   CASEB   2(R8), #0, #4                        ; 5406
 00B6      000C            0233               0233     00264 24$:   .WORD   55$-24$,-                            ;
                                              00C2     0026C                55$-24$,-                            ;
                                                                            25$-24$,-                            ;
                                                                            31$-24$,-                            ;
                                                                            33$-24$                              ;
                                                5C  11 0026E        BRB     27$                                  ;
                             0E 00000000'       EF  E8 00270 25$:   BLBS    SKIP_ACCVIOS, 26$                    ; 5430
                             07 00000000'       EF  E8 00277        BLBS    SKIP_WATCHES, 26$                    ; 5431
                             47 0C000000'       EF  E9 0027E        BLBC    DBG$GB_SET_WATCH_FLAG, 27$           ; 5432
                             59                 01  D0 00285 26$:   MOVL    #1, ACTIVE_EVENT                     ; 5441
                                         34     A6  9F 00288        PUSHAB  52(EVENT)                            ; 5454
                   00000000G 00                 01  FB 0028B        CALLS   #1, DBG$DATA_LENGTH                  ;
                             54                 50  D0 00292        MOVL    R0, LENGTH                           ;
                             50          07     A4  9E 00295        MOVAB   7(R4), R0                            ; 5455
                 54          50          FD     8F  78 00299        ASHL    #-3, R0, LENGTH                      ;
                                                56  DD 0029E        PUSHL   EVENT                                ; 5456
                      0000V  CF                 01  FB 002A0        CALLS   #1, GET_WATCH_POINT_VALUE            ;
                  00000000'  EF          30     A6  D0 002A5        MOVL    48(EVENT), OLDVALUE                  ; 5458
                             50 00000000'       EF  D0 002AD        MOVL    OLDVALUE, R0                         ; 5467
                             55 00000000'       EF  D0 002B4        MOVL    NEWVALUE, R5                         ; 5469
           20    A5          20  A0             54  29 002BB        CMPC3   LENGTH, 32(R0), 32(R5)               ;
                                                0C  12 002C1        BNEQ    28$                                  ;
```

```
                                           55  DD 002C3       PUSHL   R5                                  ; 5473
              00000000G  00                01  FB 002C5       CALLS   #1, DBG$NFREE_DESC                  ;
                                         01C8  31 002CC 27$:  BRW     55$                                 ;
                         07 00000000'      EF  E8 002CF 28$:  BLBS    SKIP_WATCHES, 29$                   ; 5480
                         08 00000000'      EF  E9 002D6       BLBC    DBG$GB_SET_WATCH_FLAG, 30$          ; 5481
              00000000'  EF 00000000'      EF  D0 002DD 29$:  MOVL    WATCH_PC, ACCVIO_PC                 ; 5483
                                      04   AC  DD 002E8 30$:  PUSHL   SIGNAL_ARG_PTR                      ; 5486
                                           56  DD 002EB       PUSHL   EVENT                               ; 5485
                  0000V  CF                02  FB 002ED       CALLS   #2, DBG$PROCESS_EVENT               ;
                     0C  AE                50  D0 002F2       MOVL    R0, STATUS                          ;
                         58 00000000'      EF  D0 002F6       MOVL    NEWVALUE, R8                        ; 5496
                         50 00000000'      EF  D0 002FD       MOVL    OLDVALUE, R0                        ; 5497
   20    A0          20  A8                54  28 00304       MOVC3   LENGTH, 32(R8), 32(R0)              ;
                                           58  DD 0030A       PUSHL   R8                                  ; 5499
              00000000G  00                01  FB 0030C       CALLS   #1, DBG$NFREE_DESC                  ;
                         B5          0C    AE  E9 00313       BLBC    STATUS, 27$                         ; 5506
                                         0197  31 00317       BRW     57$                                 ; 5508
                         5B          2C    A6  D1 0031A 31$:  CMPL    44(EVENT), USERS_PC                 ; 5524
                                           AC  12 0031E       BNEQ    27$                                 ;
                         59                01  D0 00320       MOVL    #1, ACTIVE_EVENT                    ; 5535
                                         0164  31 00323 32$:  BRW     54$                                 ; 5538
                         01                68  91 00326 33$:  CMPB    (R8), #1                            ; 5563
                                           3C  1A 00329       BGTRU   34$                                 ;
                         5B          2C    A6  D1 0032B       CMPL    44(EVENT), USERS_PC                 ; 5575
                                           9B  12 0032F       BNEQ    27$                                 ;
               00000454  8F                5A  D1 00331       CMPL    EXCEPTION_TYPE, #1108               ; 5576
                                           92  13 00338       BEQL    27$                                 ;
                         59                01  D0 0033A       MOVL    #1, ACTIVE_EVENT                    ; 5583
                                           10  DD 0033D       PUSHL   #16                                 ; 5596
              00000000G  00                01  FB 0033F       CALLS   #1, DBG$GET_MEMORY                  ;
                         57                50  D0 00346       MOVL    R0, SKIPS_ENTRY                     ;
                     04  B6                67  0E 00349       INSQUE  (SKIPS_ENTRY), @4(EVENT)            ; 5598
                     14  A7                04  B0 0034D       MOVW    #4, 20(SKIPS_ENTRY)                 ; 5607
                     16  A7                04  90 00351       MOVB    #4, 22(SKIPS_ENTRY)                 ; 5611
                     17  A7                04  88 00355       BISB2   #4, 23(SKIPS_ENTRY)                 ; 5613
                                           56  DD 00359       PUSHL   EVENT                               ; 5615
                                           57  DD 0035B       PUSHL   SKIPS_ENTRY                         ;
                  0000V  CF                02  FB 0035D       CALLS   #2, INSQUE1                         ;
                                     24    A7  D4 00362       CLRL    36(SKIPS_ENTRY)                     ; 5621
                                           4E  11 00365       BRB     38$                                 ; 5557
                         03                68  91 00367 34$:  CMPB    (R8), #3                            ; 5629
                                           4C  1F 0036A       BLSSU   39$                                 ;
                         04                68  91 0036C       CMPB    (R8), #4                            ;
                                           47  1A 0036F       BGTRU   39$                                 ;
               00000454  8F                5A  D1 00371       CMPL    EXCEPTION_TYPE, #1108               ; 5634
                                           2A  12 00378       BNEQ    37$                                 ;
                         04                6E  D1 0037A       CMPL    OPCODE, #4                          ; 5639
                                           09  12 0037D       BNEQ    35$                                 ;
                                           56  DD 0037F       PUSHL   EVENT                               ; 5641
                  0000V  CF                01  FB 00381       CALLS   #1, MATCH_RIGHT_CALL_FRAME          ;
                                           02  11 00386       BRB     36$                                 ;
                                           50  D4 00388 35$:  CLRL    RIGHT_FP_FLAG                       ; 5643
                         28                50  E9 0038A 36$:  BLBC    RIGHT_FP_FLAG, 38$                  ; 5645
                         59                01  D0 0038D       MOVL    #1, ACTIVE_EVENT                    ; 5648
         8E          19  A6                02  E1 00390       BBC     #2, 25(EVENT), 32$                  ; 5655
                     19  A6                04  8A 00395       BICB2   #4, 25(EVENT)                       ; 5658
                     01  A8                01  90 00399       MOVB    #1, 1(R8)                           ; 5659
```

```
                02  A8        1B  A6  90 0039D          MOVB    27(EVENT), 2(R8)                                ; 5660
                                  11  11 003A2          BRB     38$                                             ; 5655
          00000464  8F            5A  D1 003A4 37$:     CMPL    EXCEPTION_TYPE, #1124                           ; 5677
                                  61  12 003AB          BNEQ    45$
        53        19  A6          02  E1 003AD          BBC     #2, 25(EVENT), 43$                              ; 5678
                      59          01  D0 003B2          MOVL    #1, ACTIVE_EVENT                                ; 5679
                                00DF  31 003B5 38$:     BRW     55$                                             ; 5678
                      06          68  91 003B8 39$:     CMPB    (R8), #6                                        ; 5690
                                  F8  12 003BB          BNEQ    38$
                      24  A6 00000000G   00  D1 003BD    CMPL    DBG$RUNFRAME+56, 36(EVENT)                      ; 5697
                                  EE  12 003C5          BNEQ    38$
                      04          6E  D1 003C7          CMPL    OPCODE, #4                                      ; 5698
                                  E9  12 003CA          BNEQ    38$
                      03  A8        80  8F  88 003CC    BISB2   #128, 3(R8)                                     ; 5701
                      59          01  D0 003D1          MOVL    #1, ACTIVE_EVENT                                ; 5702
                          00000000'  EF  94 003D4       CLRB    EVENT$B_STOP_USER                               ; 5703
                          00000000G  00  94 003DA       CLRB    DBG$GB_TAKE_CMD                                 ; 5704
                                00CE  31 003E0          BRW     57$                                             ; 5700
          00000464  8F            5A  D1 003E3 40$:     CMPL    EXCEPTION_TYPE, #1124                           ; 5723
                                  09  13 003EA          BEQL    41$
          0000046C  8F            5A  D1 003EC          CMPL    EXCEPTION_TYPE, #1132
                                  07  12 003F3          BNEQ    42$
        16        19  A6          02  E1 003F5 41$:     BBC     #2, 25(EVENT), 46$                              ; 5724
                                  0E  11 003FA          BRB     44$                                             ; 5726
          00000414  8F            5A  D1 003FC 42$:     CMPL    EXCEPTION_TYPE, #1044                           ; 5733
                                  B0  12 00403          BNEQ    38$
        AB        19  A6          02  E1 00405 43$:     BBC     #2, 25(EVENT), 38$                              ; 5734
                      5B        2C  A6  D1 0040A 44$:     CMPL    44(EVENT), USERS_PC                             ; 5736
                                  A5  12 0040E 45$:     BNEQ    38$
                      59          01  D0 00410 46$:     MOVL    #1, ACTIVE_EVENT                                ; 5753
          00000414  8F            5A  D1 00413          CMPL    EXCEPTION_TYPE, #1044                           ; 5759
                                  04  12 0041A          BNEQ    47$
                19  A6            04  8A 0041C          BICB2   #4, 25(EVENT)                                   ; 5761
                          18  A6  95 00420 47$:         TSTB    24(EVENT)                                       ; 5771
                                  0D  18 00423          BGEQ    48$
          40000000  8F 00000000G  00  D1 00425          CMPL    DBG$RUNFRAME+64, #1073741824                    ; 5772
                                  65  18 00430          BGEQ    55$
                      08        02  A8  91 00432 48$:     CMPB    2(R8), #8                                       ; 5776
                                  52  13 00436          BEQL    54$
                      05        02  A8  91 00438        CMPB    2(R8), #5                                       ; 5779
                                  06  1F 0043C          BLSSU   49$
                      06        02  A8  91 0043E        CMPB    2(R8), #6
                                  06  1B 00442          BLEQU   50$
                      09        02  A8  91 00444 49$:     CMPB    2(R8), #9
                                  07  12 00448          BNEQ    51$
        48        28  B6            6E  E1 0044A 50$:     BBC     OPCODE, @40(EVENT), 55$                         ; 5788
                                  39  11 0044F          BRB     54$                                             ; 5791
                      07        02  A8  91 00451 51$:     CMPB    2(R8), #7
                                  33  12 00455          BNEQ    54$
                34  A6            5B  D1 00457          CMPL    USERS_PC, 52(EVENT)                             ; 5803
                                  06  1F 0045B          BLSSU   52$
                38  A6            5B  D1 0045D          CMPL    USERS_PC, 56(EVENT)                             ; 5805
                                  34  1B 00461          BLEQU   55$
                              10  AE  D4 00463 52$:     CLRL    MODRST                                          ; 5823
                              10  AE  9F 00466          PUSHAB  MODRST                                          ; 5829
                              38  A6  9F 00469          PUSHAB  56(EVENT)
                              34  A6  9F 0046C          PUSHAB  52(EVENT)                                       ; 5828
```

```
                          20  AE  9F 0046F        PUSHAB  STMTNO                              ; 5824
                          28  AE  9F 00472        PUSHAB  LINENO                              ;
                              5B  DD 00475        PUSHL   USERS_PC                            ; 5829
      00000000G  00               06  FB 00477        CALLS   #6, DBG$PC_TO_LINE_LOOKUP           ;
                 04           50  E8 0047E        BLBS    STATUS, 53$                         ; 5831
             34  A6           01  7D 00481        MOVQ    #1, 52(EVENT)                       ; 5833
                 02           50  D1 00485 53$:   CMPL    STATUS, #2                          ; 5853
                              0D  13 00488        BEQL    55$                                 ;
                          04  AC  DD 0048A 54$:   PUSHL   SIGNAL_ARG_PTR                      ; 5871
                              56  DD 0048D        PUSHL   EVENT                               ; 5870
          0000V  CF           02  FB 0048F        CALLS   #2, DBG$PROCESS_EVENT               ;
                 1A           50  E8 00494        BLBS    R0, 57$                             ;
      00000000G  00           00  FB 00497 55$:   CALLS   #0, DBG$REL_TEMPMEM                 ; 5894
      00000000G  00           00  FB 0049E        CALLS   #0, DBG$RST_TEMP_RELEASE            ; 5895
             08  AE           56  D1 004A5 56$:   CMPL    EVENT, LAST1                        ; 5905
                              06  13 004A9        BEQL    57$                                 ;
                 56           66  D0 004AB        MOVL    (EVENT), EVENT                      ; 5910
                            FD73  31 004AE        BRW     18$                                 ; 5352
                   00000000'  EF  7C 004B1 57$:   CLRQ    SKIP_ACCVIOS                        ; 5916
                              4A  11 004B7        BRB     62$                                 ; 5328
                 0C           5A  D1 004B9 58$:   CMPL    EXCEPTION_TYPE, #12                 ; 5921
                              45  12 004BC        BNEQ    62$                                 ;
                 3E 00000000'  EF  E8 004BE        BLBS    SKIP_ACCVIOS, 62$                   ; 5934
                 50        04  AC  D0 004C5        MOVL    SIGNAL_ARG_PTR, R0                  ; 5941
  52         0C  A0  000001FF  8F  CB 004C9        BICL3   #511, 12(R0), PAGE_ADDRESS          ;
                 50 00000000'  EF  D0 004D2        MOVL    EVENT$PAGE_QUEUE, PAGE_ENTRY        ; 5942
                 51 00000000'  EF  9E 004D9 59$:   MOVAB   EVENT$PAGE_QUEUE, R1                ; 5947
                 51           50  D1 004E0        CMPL    PAGE_ENTRY, R1                      ;
                              17  13 004E3        BEQL    61$                                 ;
                 52        08  A0  D1 004E5        CMPL    8(PAGE_ENTRY), PAGE_ADDRESS         ; 5949
                              0C  12 004E9        BNEQ    60$                                 ;
      00000000'  EF           01  D0 004EB        MOVL    #1, SKIP_ACCVIOS                    ; 5957
                 59           01  D0 004F2        MOVL    #1, ACTIVE_EVENT                    ; 5958
                              05  11 004F5        BRB     61$                                 ; 5956
                 50           60  D0 004F7 60$:   MOVL    (PAGE_ENTRY), PAGE_ENTRY            ; 5963
                              DD  11 004FA        BRB     59$                                 ; 5947
      00000000'  EF           5B  D0 004FC 61$:   MOVL    USERS_PC, ACCVIO_PC                 ; 5971
                 42           59  E8 00503 62$:   BLBS    ACTIVE_EVENT, 66$                   ; 5983
                 56 00000000'  EF  D0 00506        MOVL    EVENT$CMD_QUEUE, EVENT              ; 5990
                 50 00000000'  EF  9E 0050D 63$:   MOVAB   EVENT$CMD_QUEUE, R0                 ; 5991
                 50           56  D1 00514        CMPL    EVENT, R0                           ;
                              2C  13 00517        BEQL    65$                                 ;
                          17  A6  95 00519        TSTB    23(EVENT)                           ; 5997
                              1C  19 0051C        BLSS    64$                                 ;
                 02        15  A6  91 0051E        CMPB    21(EVENT), #2                       ; 6003
                              16  12 00522        BNEQ    64$                                 ;
                 59           01  D0 00524        MOVL    #1, ACTIVE_EVENT                    ; 6010
      00000000G  00           01  90 00527        MOVB    #1, DBG$GB_EXC_BRE_FLAG             ; 6015
                          04  AC  DD 0052E        PUSHL   SIGNAL_ARG_PTR                      ; 6020
                              56  DD 00531        PUSHL   EVENT                               ;
          0000V  CF           02  FB 00533        CALLS   #2, DBG$PROCESS_EVENT               ;
                              0B  11 00538        BRB     65$                                 ; 6005
             08  AE           56  D1 0053A 64$:   CMPL    EVENT, LAST1                        ; 6029
                              05  13 0053E        BEQL    65$                                 ;
                 56           66  D0 00540        MOVL    (EVENT), EVENT                      ; 6034
                              C8  11 00543        BRB     63$                                 ; 5991
                 6C           59  E9 00545 65$:   BLBC    ACTIVE_EVENT, 72$                   ; 6048
```

```
                          08 00000000'   EF   E8 00548 66$:    BLBS    EVENT$B_STOP_USER, 67$                          ; 6059
                       35 00000000G 00   01   E1 0054F         BBC     #1, DBG$GV_CONTROL+1, 70$                       ; 6060
                          00000000G 00        8F   8A 00557 67$:    BICB2   #128, DBG$GL_OUTPRAB+7                      ; 6070
                          02 00000000G 00   91   0055F         CMPB    DBG$GB_CALL_NORMAL_RET, #2                      ; 6080
                                        07   13   00566         BEQL    68$                                             ;
                             00000000G 00   08   88   00568    BISB2   #8, DBG$GV_CONTROL+1                            ; 6082
                          16 00000000G 00   E9   0056F 68$:    BLBC    DBG$GB_TAKE_CMD, 70$                            ; 6084
                             00000000G 00   94   00576         CLRB    DBG$GB_CALL_NORMAL_RET                          ;
                          09 00000000G 00   E9   0057C 69$:    BLBC    DBG$GB_TAKE_CMD, 70$                            ; 6093
                             00000000G 00   FB   00583         CALLS   #0, DBG$COMMAND_PROC                            ; 6094
                                        F0   11   0058A         BRB     69$                                             ;
                                F56D    CF   00   FB 0058C 70$:    CALLS   #0, DBG$ACTIVATE_EVENTS                         ; 6101
                          18 00000000G 00   E9   00591         BLBC    DBG$GB_EXC_BRE_FLAG, 71$                        ; 6107
00000000G  00        01          04     04   AE   F0 00598         INSV    ORIG_TBIT, #4, #1, DBG$RUNFRAME+68              ; 6110
                          1A 00000000G 00   E8   005A2         BLBS    DBG$GB_TAKE_CMD, 73$                            ; 6117
                          13 00000000G 00   E9   005A9         BLBC    DBG$GB_GO_ARG_FLAG, 73$                         ; 6119
                          50             01   D0   005B0 71$:    MOVL    #1, R0                                          ; 6132
                                              04   005B3         RET                                                     ;
                                F545    CF   00   FB 005B4 72$:    CALLS   #0, DBG$ACTIVATE_EVENTS                         ; 6138
00000000G  00        01          04     04   AE   F0 005B9         INSV    ORIG_TBIT, #4, #1, DBG$RUNFRAME+68              ; 6139
                          50       0918  8F   3C   005C3 73$:    MOVZWL  #2328, R0                                       ; 6140
                                              04   005C8         RET                                                     ; 6141

; Routine Size:  1481 bytes,    Routine Base:  DBG$CODE + 1E6F
```

```
;  6026     6142  1 GLOBAL ROUTINE DBG$PROCESS_EVENT(EVENT, SIGNAL_ARG_PTR) =
;  6027     6143  1 !
;  6028     6144  1 !++
;  6029     6145  1 ! FUNCTIONAL DESCRIPTION:
;  6030     6146  1 !
;  6031     6147  1 !       An eventpoint has been determined to be active.  It must now be
;  6032     6148  1 !       processed.  This includes handling any /AFTER, WHEN, and DO.
;  6033     6149  1 !
;  6034     6150  1 ! FORMAL PARAMETERS:
;  6035     6151  1 !
;  6036     6152  1 !       EVENT : The address of the event entry.
;  6037     6153  1 !
;  6038     6154  1 ! IMPLICIT OUTPUTS:
;  6039     6155  1 !
;  6040     6156  1 !       Processing the event may involve the execution of user commands.
;  6041     6157  1 !
;  6042     6158  1 ! ROUTINE VALUE:
;  6043     6159  1 !
;  6044     6160  1 !       TRUE is a GO, STEP, CONTINUE, or EXIT command was executed within
;  6045     6161  1 !       a DO clause.
;  6046     6162  1 !
;  6047     6163  1 ! SIDE EFFECTS:
;  6048     6164  1 !
;  6049     6165  1 !       Any number of things.
;  6050     6166  1 !--
;  6051     6167  1
;  6052     6168  2     BEGIN
;  6053     6169  2
;  6054     6170  2     MAP
;  6055     6171  2         EVENT :             REF EVENT$EVENT_DESCRIPTOR, ! Event entry pointer
;  6056     6172  2         SIGNAL_ARG_PTR :    REF BLOCK[,BYTE];
;  6057     6173  2
;  6058     6174  2     BUILTIN CALLG;
;  6059     6175  2     EXTERNAL ROUTINE dbg$final_handl;
;  6060     6176  2
;  6061     6177  2     LOCAL
;  6062     6178  2         DO_LIST_ENTRY : REF EVENT$DO_LIST_DESCRIPTOR, ! DO LIST descriptor
;  6063     6179  2         MESSAGE_VECT,             ! Message Vector
;  6064     6180  2         SAVED_CISHEAD,            ! Saved pointer to DBG$GL_CISHEAD
;  6065     6181  2         SAVED_LEVEL,              ! Saved level of CIS nesting
;  6066     6182  2         EXCEPTION_TYPE;
;  6067     6183  2
;  6068     6184  2     ENABLE dbg$final_handl;
;  6069     6185  2
;  6070     6186  2     ! Save the exception type (name);
;  6071     6187  2     !
;  6072     6188  2     EXCEPTION_TYPE = .SIGNAL_ARG_PTR [CHF$L_SIG_NAME];
;  6073     6189  2
;  6074     6190  2     ! If this is a skips-entry, 'delete' this one and point to the parent.
;  6075     6191  2     ! It is the one we want to actually process.
;  6076     6192  2     !
;  6077     6193  2     IF .EVENT [EVENT$B_CMD_TYPE] EQLU EVENT$K_TYPE_SKIPS
;  6078     6194  2     THEN
```

```
;  6083          6199  3          BEGIN
;  6084          6200  3          EVENT [EVENT$V_DELETED] = TRUE;
;  6085          6201  3          WHILE .EVENT [EVENT$B_CMD_TYPE] EQLU EVENT$K_TYPE_SKIPS DO
;  6086          6202  3              EVENT = .EVENT [EVENT$L_EXC_FLINK];
;  6087          6203  3          END;
;  6088          6204
;  6089          6205
;  6090          6206       ! If this is a steps-event, process it differently from the others.
;  6091          6207       !
;  6092          6208       IF .EVENT [EVENT$B_CMD_TYPE] EQLU EVENT$K_TYPE_STEPS
;  6093          6209       THEN
;  6094          6210           BEGIN
;  6095          6211
;  6096          6212
;  6097          6213           ! Decrement the step (after) count.
;  6098          6214           !
;  6099          6215           IF ((EVENT [EVENT$L_AFTER_COUNT] =
;  6100          6216                .EVENT [EVENT$L_AFTER_COUNT] - 1
;  6101          6217               ) EQLU 0
;  6102          6218              )
;  6103          6219           THEN
;  6104          6220
;  6105          6221
;  6106          6222               ! The stepping is complete....
;  6107          6223               !
;  6108          6224               BEGIN
;  6109          6225  4
;  6110          6226  4            ! Special case for STEP/EXCEPTION.
;  6111          6227  4            !
;  6112          6228  4            IF .EVENT[EVENT$B_CMD_KIND] EQL EVENT$K_KIND_EXC
;  6113          6229  4            THEN
;  6114          6230  5                BEGIN
;  6115          6231  5
;  6116          6232  5
;  6117          6233  5                ! This is an exception break or trace, so flag it.
;  6118          6234  5                !
;  6119          6235  5                DBG$GB_EXC_BRE_FLAG = TRUE;
;  6120          6236  5
;  6121          6237  5
;  6122          6238  5                ! Format and output the message.
;  6123          6239  5                !
;  6124          6240  5                DBG$PUTMSG (.SIGNAL_ARG_PTR);
;  6125          6241  4                END;
;  6126          6242  4
;  6127          6243  4            ! Announce the step.
;  6128          6244  4            !
;  6129          6245  4            ANNOUNCE_EVENT (.EVENT, .EXCEPTION_TYPE);
;  6130          6246  4
;  6131          6247  4
;  6132          6248  4            ! Delete the step.
;  6133          6249  4            !
;  6134          6250  4            EVENT [EVENT$V_DELETED] = TRUE;
;  6135          6251  4
;  6136          6252  4
;  6137          6253  4            ! DEBUG will stop for commands.
;  6138          6254  4            !
;  6139          6255  4            EVENT$B_STOP_USER = TRUE;
```

```
;  6140          6256  3              END;
;  6141          6257  3
;  6142          6258  3
;  6143          6259  3          ! Return - we haven't executed any new commands....
;  6144          6260  3          !
;  6145          6261  3          RETURN FALSE;
;  6146          6262  2          END;
;  6147          6263  2
;  6148          6264  2
;  6149          6265  2      ! This is not a steps (or skips) entry.  Process normally.
;  6150          6266  2      ! Update the /AFTER count.  If it becomes equal to 0, then the entry is
;  6151          6267  2      ! now active.
;  6152          6268  2      !
;  6153          6269  4      IF ((EVENT [EVENT$L_AFTER_COUNT] =
;  6154          6270  4           .EVENT [EVENT$L_AFTER_COUNT] - 1
;  6155          6271  3          ) EQLU 0
;  6156          6272  3         )
;  6157          6273  2      THEN
;  6158          6274  2          BEGIN
;  6159          6275  3
;  6160          6276  3          ! We've passed the /AFTER test.  Now set the /AFTER count to 1.
;  6161          6277  3          !
;  6162          6278  3          EVENT [EVENT$L_AFTER_COUNT] = 1;
;  6163          6279  3
;  6164          6280  3
;  6165          6281  3          ! If there is a WHEN expression, procees it....
;  6166          6282  3          !
;  6167          6283  3          IF .EVENT [EVENT$L_WHEN] NEQA 0
;  6168          6284  3          THEN
;  6169          6285  3              BEGIN
;  6170          6286  4              PARSE_WHEN_CONDITION(.EVENT);
;  6171          6287  4
;  6172          6288  4
;  6173          6289  4              ! If the condition is false, return FALSE.
;  6174          6290  4              !
;  6175          6291  4              IF NOT .WHEN_CONDITION
;  6176          6292  4              THEN
;  6177          6293  4                  RETURN FALSE;
;  6178          6294  4              END;
;  6179          6295  3
;  6180          6296  3
;  6181          6297  3          ! If this event is temporary, flag it as deleted now.  This shouldn't
;  6182          6298  3          ! affect any further processing, but the entry will be deleted when
;  6183          6299  3          ! we go off to the user program.
;  6184          6300  3          !
;  6185          6301  3          IF .EVENT [EVENT$V_ONCE_ONLY]
;  6186          6302  3          THEN
;  6187          6303  3              EVENT [EVENT$V_DELETED] = TRUE;
;  6188          6304  3
;  6189          6305  3
;  6190          6306  3          ! Flag a stop, if this is a BREAK.
;  6191          6307  3          !
;  6192          6308  3          IF .EVENT [EVENT$B_CMD_TYPE] EQLU EVENT$K_TYPE_BREAK OR
;  6193          6309  3             .EVENT [EVENT$B_CMD_TYPE] EQLU EVENT$K_TYPE_WATCH
;  6194          6310  3          THEN
;  6195          6311  3              EVENT$B_STOP_USER = TRUE;
;  6196          6312  3
```

```
;  6197          6313  3
;  6198          6314  3
;  6199          6315  3          ! Case on the kind of event entry, to determine how to process it.
;  6200          6316  3          !
;  6201          6317  3          CASE .EVENT [EVENT$B_CMD_KIND]
;  6202          6318  3          FROM EVENT$K_KIND_ACC TO EVENT$K_KIND_EXC OF
;  6203          6319  3              SET
;  6204          6320  3              [EVENT$K_KIND_ACC]:
;  6205          6321  3                  ANNOUNCE_EVENT (.EVENT, .EXCEPTION_TYPE);
;  6206          6322  3
;  6207          6323  3              [EVENT$K_KIND_INS]:
;  6208          6324  3                  ANNOUNCE_EVENT (.EVENT, .EXCEPTION_TYPE);
;  6209          6325  3
;  6210          6326  3              [EVENT$K_KIND_EXC]:
;  6211          6327  4                  BEGIN
;  6212          6328  4
;  6213          6329  4
;  6214          6330  4                  ! This is an exception break or trace, so flag it.
;  6215          6331  4                  !
;  6216          6332  4                  DBG$GB_EXC_BRE_FLAG = TRUE;
;  6217          6333  4
;  6218          6334  4
;  6219          6335  4                  ! Format and output the message.
;  6220          6336  4                  !
;  6221          6337  4                  DBG$PUTMSG (.SIGNAL_ARG_PTR);
;  6222          6338  4
;  6223          6339  4
;  6224          6340  4                  ! Announce the break.
;  6225          6341  4                  !
;  6226          6342  4                  ANNOUNCE_EVENT (.EVENT, .EXCEPTION_TYPE);
;  6227          6343  3                  END;
;  6228          6344  3              TES;
;  6229          6345  3
;  6230          6346  3
;  6231          6347  3          ! If there is a DO associated with this entry, do it.
;  6232          6348  3          !
;  6233          6349  3          IF .EVENT [EVENT$L_DO] NEQA 0
;  6234          6350  3          THEN
;  6235          6351  4              BEGIN
;  6236          6352  4
;  6237          6353  4
;  6238          6354  4              ! We need a local pointer to the DO command string.
;  6239          6355  4              !
;  6240          6356  4              LOCAL
;  6241          6357  4                  DO_LENGTH,
;  6242          6358  4                  DO_STRING :   REF VECTOR [, BYTE];
;  6243          6359  4
;  6244          6360  4
;  6245          6361  4              ! Point to the DO LIST entry.
;  6246          6362  4              !
;  6247          6363  4              DO_LIST_ENTRY = .EVENT [EVENT$L_DO];
;  6248          6364  4
;  6249          6365  4
;  6250          6366  4              ! Point to the command string.
;  6251          6367  4              !
;  6252          6368  4              DO_LENGTH = .(.DO_LIST_ENTRY[EVENT$L_DO_LIST_POINT])<0,16,0>;
;  6253          6369  4
```

```
;  6254          6370  4          DO_STRING = DBG$GET_MEMORY((.DO_LENGTH/%UPVAL)+1);
;  6255          6371  4          CH$MOVE(.DO_LENGTH,.DO_LIST_ENTRY[EVENT$L_DO_LIST_POINT]+2,.DO_STRING);
;  6256          6372  4          DO_STRING[.DO_LENGTH] = 0;
;  6257          6373  4
;  6258          6374  4
;  6259          6375  4          ! Add a link to the command input stream. First remember the
;  6260          6376  4          ! current pointer to the head of the CIS stack, and the number
;  6261          6377  4          ! of levels of nesting.
;  6262          6378  4          !
;  6263          6379  4          SAVED_CISHEAD = .DBG$GL_CISHEAD;
;  6264          6380  4          SAVED_LEVEL = .DBG$GL_CIS_LEVELS;
;  6265          6381  4          IF NOT DBG$NCIS_ADD (.DO_STRING, .DO_LENGTH, CIS_INPBUF,
;  6266          6382  4                                                  0, 0, MESSAGE_VECT)
;  6267          6383  4          THEN
;  6268          6384  4              CALLG (.MESSAGE_VECT, LIB$SIGNAL);
;  6269          6385  4
;  6270          6386  4
;  6271          6387  4          ! Process the DO clause. We repeatedly call COMMAND_PROC, each
;  6272          6388  4          ! call executing one command from the DO list. We exit the
;  6273          6389  4          ! loop if either: 1) the user said STEP or GO, thus resulting
;  6274          6390  4          ! in the TAKE_CMD flag becoming false, or 2) We exhaust the
;  6275          6391  4          ! commands in the current CIS, and get back to the SAVED_CISHEAD
;  6276          6392  4          ! level.
;  6277          6393  4          !
;  6278          6394  4          DBG$NINITIALIZE ();
;  6279          6395  4          WHILE (.DBG$GB_TAKE_CMD) AND
;  6280          6396  4                (.DBG$GL_CISHEAD NEQ .SAVED_CISHEAD) AND
;  6281          6397  4                (.DBG$GL_CIS_LEVELS GTR .SAVED_LEVEL) DO
;  6282          6398  4              DBG$COMMAND_PROC ();
;  6283          6399  4
;  6284          6400  4
;  6285          6401  4          ! If the user typed a GO, STEP, EXIT, or CONTINUE, return true,
;  6286          6402  4          ! otherwise false.
;  6287          6403  4          !
;  6288          6404  4          RETURN NOT .DBG$GB_TAKE_CMD;
;  6289          6405  3          END;
;  6290          6406  2      END;
;  6291          6407  2
;  6292          6408  2
;  6293          6409  2      ! Return FALSE since the /AFTER count hasn't run out.
;  6294          6410  2      !
;  6295          6411  2      RETURN FALSE;
;  6296          6412  1  END;


                                    0FFC 00000          .ENTRY  DBG$PROCESS_EVENT, Save R2,R3,R4,R5,R6,R7,- ; 6142
                                                                R8,R9,R10,R11
5B 00000000G   00   9E 00002        MOVAB   DBG$GL_CIS_LEVELS, R11                              :
5A 00000000G   00   9E 00009        MOVAB   DBG$GL_CISHEAD, R10                                 :
59 00000000'   EF   9E 00010        MOVAB   EVENT$B_STOP_USER, R9                               :
58 00000000G   00   9E 00017        MOVAB   DBG$PUTMSG, R8                                      :
5E             04   C2 0001E        SUBL2   #4, SP                                              :
6D      012A   CF   DE 00021        MOVAL   175, (FP)                                           ; 6170
54        08   AC   D0 00026        MOVL    SIGNAL_ARG_PTR, R4                                  ; 6191
```

```
                        55    04    A4   D0 0002A          MOVL    4(R4), EXCEPTION_TYPE                   ;
                        50    04    AC   D0 0002E          MOVL    EVENT, R0                               ; 6197
                        04    14    A0   91 00032          CMPB    20(R0), #4                              ;
                                    16   12 00036          BNEQ    2$                                      ;
                    17  AC    80    8F   88 00038          BISB2   #128, 23(R0)                            ; 6200
                        50    04    AC   D0 0003D 1$:      MOVL    EVENT, R0                               ; 6201
                        04    14    A0   91 00041          CMPB    20(R0), #4                              ;
                                    07   12 00045          BNEQ    2$                                      ;
                    04  AC    08    A0   D0 00047          MOVL    8(R0), EVENT                            ; 6202
                                    EF   11 0004C          BRB     1$                                      ;
                        52    04    AC   D0 0004E 2$:      MOVL    EVENT, R2                               ; 6208
                        53    14    A2   9E 00052          MOVAB   20(R2), R3                              ;
                        03          63   91 00056          CMPB    (R3), #3                                ;
                                    29   12 00059          BNEQ    5$                                      ;
                              1C    A2   D7 0005B          DECL    28(R2)                                  ; 6216
                                    21   12 0005E          BNEQ    4$                                      ; 6217
                        02    01    A3   91 00060          CMPB    1(R3), #2                               ; 6228
                                    0C   12 00064          BNEQ    3$                                      ;
              00000000G     00          01   90 00066          MOVB    #1, DBG$GB_EXC_BRE_FLAG                 ; 6235
                                    54   DD 0006D          PUSHL   R4                                      ; 6240
                        68          01   FB 0006F          CALLS   #1, DBG$PUTMSG                          ;
                                    24   BB 00072 3$:      PUSHR   #^M<R2,R5>                              ; 6245
                0000V   CF          02   FB 00074          CALLS   #2, ANNOUNCE_EVENT                      ;
                  03    A3    80    8F   88 00079          BISB2   #128, 3(R3)                             ; 6250
                        69          01   90 0007E          MOVB    #1, EVENT$B_STOP_USER                   ; 6255
                                00C8  31   00081 4$:       BRW     16$                                     ; 6261
                              1C    A2   D7 00084 5$:      DECL    28(R2)                                  ; 6270
                                    F8   12 00087          BNEQ    4$                                      ; 6271
                            1C  A2        01   D0 00089          MOVL    #1, 28(R2)                      ; 6279
                              20    A2   D5 0008D          TSTL    32(R2)                                  ; 6284
                                    0E   13 00090          BEQL    6$                                      ;
                                    52   DD 00092          PUSHL   R2                                      ; 6287
                0000V   CF          01   FB 00094          CALLS   #1, PARSE_WHEN_CONDITION                ;
                        E1  00000000' EF   E9 00099          BLBC    WHEN_CONDITION, 4$                      ; 6292
              05            63          1A   E1 000A0 6$:      BBC     #26, (R3), 7$                       ; 6302
                    03  A3        80    8F   88 000A4          BISB2   #128, 3(R3)                         ; 6304
                                    63   95 000A9 7$:      TSTB    (R3)                                    ; 6309
                                    05   13 000AB          BEQL    8$                                      ;
                        02          63   91 000AD          CMPB    (R3), #2                                ; 6310
                                    03   12 000B0          BNEQ    9$                                      ;
                        69          01   90 000B2 8$:      MOVB    #1, EVENT$B_STOP_USER                   ; 6312
              02            00    01    A3   8F 000B5 9$:      CASEB   1(R3), #0, #2                       ; 6317
            0008        0014        0014       000BA 10$:     .WORD   12$-10$,-                           ;
                                                              12$-10$,-                                   ;
                                                              11$-10$                                     ;
                                    0C   11 000C0          BRB     12$                                     ; 6324
          00000000G     00          01   90 000C2 11$:      MOVB    #1, DBG$GB_EXC_BRE_FLAG                 ; 6332
                                    54   DD 000C9          PUSHL   R4                                      ; 6337
                        68          01   FB 000CB          CALLS   #1, DBG$PUTMSG                          ;
                                    24   BB 000CE 12$:      PUSHR   #^M<R2,R5>                              ; 6342
                0000V   CF          02   FB 000D0          CALLS   #2, ANNOUNCE_EVENT                      ;
                              24    A2   D5 000D5          TSTL    36(R2)                                  ; 6349
                                    72   13 000D8          BEQL    16$                                     ;
                        50    24    A2   D0 000DA          MOVL    36(R2), DO_LIST_ENTRY                   ; 6363
                        52    0C    A0   D0 000DE          MOVL    12(DO_LIST_ENTRY), R2                   ; 6368
                        56          62   3C 000E2          MOVZWL  (R2), DO_LENGTH                         ;
              50            56          04   C7 000E5          DIVL3   #4, DO_LENGTH, R0                   ; 6370
```

```
                                 01    A0  9F 000E9         PUSHAB  1(R0)                                  ;
             00000000G  00             01  FB 000EC         CALLS   #1, DBG$GET_MEMORY                     ;
                        57             50  D0 000F3         MOVL    R0, DO_STRING                          ;
      67           02   A2             56  28 000F6         MOVC3   DO_LENGTH, 2(R2), (DO_STRING)          ; 6371
                                     6647  94 000FB         CLRB    (DO_LENGTH)[DO_STRING]                 ; 6372
                        53             6A  D0 000FE         MOVL    DBG$GL_CISHEAD, SAVED_CISHEAD          ; 6379
                        52             6B  D0 00101         MOVL    DBG$GL_CIS_LEVELS, SAVED_LEVEL         ; 6380
                                       5E  DD 00104         PUSHL   SP                                     ; 6381
                                       7E  7C 00106         CLRQ    -(SP)                                  ;
                                       02  DD 00108         PUSHL   #2                                     ;
                                       56  DD 0010A         PUSHL   DO_LENGTH                              ;
                                       57  DD 0010C         PUSHL   DO_STRING                              ;
             00000000G  00             06  FB 0010E         CALLS   #6, DBG$NCIS_ADD                       ;
                        08             50  E8 00115         BLBS    R0, 13$                                ;
             00000000G  00       00    BE  FA 00118         CALLG   @MESSAGE_VECT, LIB$SIGNAL              ; 6384
             00000000G  00             00  FB 00120  13$:   CALLS   #0, DBG$NINITIALIZE                    ; 6394
                        13 00000000G   00  E9 00127  14$:   BLBC    DBG$GB_TAKE_CMD, 15$                   ; 6395
                        53             6A  D1 0012E         CMPL    DBG$GL_CISHEAD, SAVED_CISHEAD          ; 6396
                                       0E  13 00131         BEQL    15$                                    ;
                        52             6B  D1 00133         CMPL    DBG$GL_CIS_LEVELS, SAVED_LEVEL         ; 6397
                                       09  15 00136         BLEQ    15$                                    ;
             00000000G  00             00  FB 00138         CALLS   #0, DBG$COMMAND_PROC                   ; 6398
                                       E6  11 0013F         BRB     14$                                    ;
                        50 00000000G   00  9A 00141  15$:   MOVZBL  DBG$GB_TAKE_CMD, R0                    ; 6404
                        50             50  D2 00148         MCOML   R0, R0                                 ;
                                           04 0014B         RET                                            ;
                                       50  D4 0014C  16$:   CLRL    R0                                     ; 6412
                                           04 0014E         RET                                            ;
                                     0000     0014F  17$:   .WORD   Save nothing                           ; 6170
                                       7E  D4 00151         CLRL    -(SP)                                  ;
                                       5E  DD 00153         PUSHL   SP                                     ;
                        7E       04    AC  7D 00155         MOVQ    4(AP), -(SP)                           ;
             00000000G  00             03  FB 00159         CALLS   #3, DBG$FINAL_HANDL                    ;
                                           04 00160         RET                                            ;

; Routine Size:  353 bytes,    Routine Base:  DBG$CODE + 2438
```

```
;  6298          6413  1 GLOBAL ROUTINE DBG$UPDATE_WATCHPOINTS: NOVALUE =
;  6299          6414  1 !
;  6300          6415  1 !++
;  6301          6416  1 ! FUNCTION
;  6302          6417  1 !       This routine updates the values in the watch point events.  This
;  6303          6418  1 !       is done in order to keep the watched values current even after a
;  6304          6419  1 !       DEPOSIT.
;  6305          6420  1 !
;  6306          6421  1 ! INPUTS
;  6307          6422  1 !       NONE
;  6308          6423  1 !
;  6309          6424  1 ! OUTPUTS
;  6310          6425  1 !       All watch point event entries are updated with their current value.
;  6311          6426  1 !
;  6312          6427  2     BEGIN
;  6313          6428  2
;  6314          6429  2     LOCAL
;  6315          6430  2         EVENT_ENTRY     : REF EVENT$EVENT_DESCRIPTOR,
;  6316          6431  2         LENGTH,
;  6317          6432  2         NEW_VALDESC     : REF DBG$VALDESC,
;  6318          6433  2         OLD_VALDESC     : REF DBG$VALDESC;
;  6319          6434  2
;  6320          6435  2
;  6321          6436  2     ! Starting at the head of the command queue, update each watch point
;  6322          6437  2     ! entry.
;  6323          6438  2     !
;  6324          6439  2     EVENT_ENTRY = .EVENT$CMD_QUEUE [L_QUEUE_FLINK];
;  6325          6440  2     WHILE .EVENT_ENTRY NEQA EVENT$CMD_QUEUE DO
;  6326          6441  3         BEGIN
;  6327          6442  3
;  6328          6443  3         ! If this is a watch point entry, update it.
;  6329          6444  3         !
;  6330          6445  3         IF .EVENT_ENTRY[EVENT$B_SUB_KIND] EQL EVENT$K_ACC_MDFY
;  6331          6446  3         THEN
;  6332          6447  4             BEGIN
;  6333          6448  4
;  6334          6449  4             ! Get the byte length of the entry's value.  Note that we actually
;  6335          6450  4             ! get it's bit length and round up.
;  6336          6451  4             !
;  6337          6452  4             LENGTH = DBG$DATA_LENGTH(EVENT_ENTRY[EVENT$A_VMSDESC]);
;  6338          6453  4             LENGTH = (.LENGTH + 7) ^ -3;
;  6339          6454  4
;  6340          6455  4             ! Make a new value descriptor from the VMS descriptor in this
;  6341          6456  4             ! event entry.  This will then contain the new current value
;  6342          6457  4             ! we are looking for.
;  6343          6458  4             !
;  6344          6459  4             NEW_VALDESC = DBG$MAKE_VAL_DESC(EVENT_ENTRY[EVENT$A_VMSDESC],
;  6345          6460  4                                             DBG$K_VALUE_DESC);
;  6346          6461  4
;  6347          6462  4             ! Now copy the "new" value into the old value descriptor that
;  6348          6463  4             ! hangs of this event entry.
;  6349          6464  4             !
;  6350          6465  4             OLD_VALDESC = .EVENT_ENTRY[EVENT$L_VALDESC];
;  6351          6466  4             CH$MOVE(.LENGTH,
;  6352          6467  4                     NEW_VALDESC [DBG$A_VALUE_ADDRESS],
;  6353          6468  4                     OLD_VALDESC [DBG$A_VALUE_ADDRESS]);
;  6354          6469  3             END;
```

```
;  6355          6470  !
;  6356          6471  !       Point to the next Command Queue entries.
;  6357          6472  !
;  6358          6473          EVENT_ENTRY = .EVENT_ENTRY [EVENT$L_CMD_FLINK];
;  6359          6474          END;
;  6360          6475      END;


                                                07FC 00000          .ENTRY  DBG$UPDATE_WATCHPOINTS, Save R2,R3,R4,R5,-      ; 6413
                                                                            R6,R7,R8,R9,R10
                          5A 00000000'  EF  9E 00002                MOVAB   EVENT$CMD_QUEUE, R10                            ;
                          56            6A  D0 00009                MOVL    EVENT$CMD_QUEUE, EVENT_ENTRY                    ; 6439
                          50            6A  9E 0000C 1$:            MOVAB   EVENT$CMD_QUEUE, R0                             ; 6440
                          50            56  D1 0000F                CMPL    EVENT_ENTRY, R0                                 ;
                                        3C  13 00012                BEQL    3$                                              ;
                          02      16    A6  91 00014                CMPB    22(EVENT_ENTRY), #2                             ; 6445
                                        31  12 00018                BNEQ    2$                                              ;
                                  34    A6  9F 0001A                PUSHAB  52(EVENT_ENTRY)                                 ; 6452
                00000000G 00            01  FB 0001D                CALLS   #1, DBG$DATA_LENGTH                             ;
                          59            50  D0 00024                MOVL    R0, LENGTH                                      ;
                          50      07    A9  9E 00027                MOVAB   7(R9), R0                                       ; 6453
                59        50      FD    8F  78 0002B                ASHL    #-3, R0, LENGTH                                 ;
                          7E      7A    8F  9A 00030                MOVZBL  #122, -(SP)                                     ; 6459
                                  34    A6  9F 00034                PUSHAB  52(EVENT_ENTRY)                                 ;
                00000000G 00            02  FB 00037                CALLS   #2, DBG$MAKE_VAL_DESC                           ;
                          58            50  D0 0003E                MOVL    R0, NEW_VALDESC                                 ;
                          57      30    A6  D0 00041                MOVL    48(EVENT_ENTRY), OLD_VALDESC                    ; 6465
          20    A7        20    A8      59  28 00045                MOVC3   LENGTH, 32(NEW_VALDESC), 32(OLD_VALDESC)        ; 6468
                          56            66  D0 0004B 2$:            MOVL    (EVENT_ENTRY), EVENT_ENTRY                      ; 6473
                                        BC  11 0004E                BRB     1$                                              ; 6440
                                            04 00050 3$:            RET                                                     ; 6475

; Routine Size:  81 bytes,    Routine Base:  DBG$CODE + 2599
```

```
;  6362     6476  1 ROUTINE ANNOUNCE_EVENT(EVENT_ENTRY, EXCEPTION_TYPE): NOVALUE =
;  6363     6477  1 !
;  6364     6478  1 ! FUNCTION
;  6365     6479  1 !       This routine announces an event entry if the /SILENT flag is false.
;  6366     6480  1 !
;  6367     6481  1 ! INPUTS
;  6368     6482  1 !       EVENT_ENTRY :           Pointer to event entry.
;  6369     6483  1 !
;  6370     6484  1 ! OUTPUTS
;  6371     6485  1 !       Announces the event entry via DBG$PRINT.
;  6372     6486  1 !
;  6373     6487  1
;  6374     6488  2     BEGIN
;  6375     6489  2
;  6376     6490  2     MAP
;  6377     6491  2         EVENT_ENTRY :       REF EVENT$EVENT_DESCRIPTOR, ! Event Entry Pointer
;  6378     6492  2         EXCEPTION_TYPE :    BLOCK [%UPVAL, BYTE];
;  6379     6493  2
;  6380     6494  2
;  6381     6495  2     LOCAL
;  6382     6496  2         VMS_DESC: DBG$STG_DESC,
;  6383     6497  2         VAL_DESC: REF DBG$VALDESC,
;  6384     6498  2         STRING_DESCRIPTOR : BLOCK [8, BYTE],
;  6385     6499  2         OUTPUT_BUFFER : VECTOR [TTY_OUT_WIDTH, BYTE],
;  6386     6500  2         DUMMY,
;  6387     6501  2
;  6388     6502  2         ADDRESS;                                    ! Entry address
;  6389     6503  2
;  6390     6504  2
;  6391     6505  2     ! If the /SILENT flag is set, return without announcing.
;  6392     6506  2     ! (This can be set by SET BREA/SILENT or STEP/SILENT).
;  6393     6507  2     !
;  6394     6508  2     IF .EVENT_ENTRY[EVENT$V_SILENT] THEN RETURN;
;  6395     6509  2
;  6396     6510  2
;  6397     6511  2     ! Purge type ahead.
;  6398     6512  2     !
;  6399     6513  2     DBG$GL_OUTPRAB [RAB$V_CCO] = TRUE;
;  6400     6514  2
;  6401     6515  2
;  6402     6516  2     ! Setup a string descriptor.
;  6403     6517  2     !
;  6404     6518  2     STRING_DESCRIPTOR [DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  6405     6519  2     STRING_DESCRIPTOR [DSC$B_CLASS] = DSC$K_CLASS_S;
;  6406     6520  2     STRING_DESCRIPTOR [DSC$A_POINTER] = 0;
;  6407     6521  2     STRING_DESCRIPTOR [DSC$W_LENGTH] = 0;
;  6408     6522  2
;  6409     6523  2
;  6410     6524  2     ! Print the event type (break, trace, watch or stepto - point).
;  6411     6525  2     !
;  6412     6526  2     DBG$PRINT   ($AC    ('!AC'),
;  6413     6527  2                  SELECTONE .EVENT_ENTRY [EVENT$B_CMD_TYPE] OF
;  6414     6528  2                      SET
;  6415     6529  2                      [EVENT$K_TYPE_BREAK]:
;  6416     6530  2                          $AC ('break');
;  6417     6531  2                      [EVENT$K_TYPE_TRACE]:
;  6418     6532  2                          $AC ('trace');
```

```
;  6419          6533  2                    [EVENT$K_TYPE_WATCH]:
;  6420          6534  2                        $AC ('watch');
;  6421          6535  2                    [EVENT$K_TYPE_STEPS]:
;  6422          6536  2                        $AC ('stepped');
;  6423          6537  2                    TES
;  6424          6538  2                );
;  6425          6539  2
;  6426          6540  2
;  6427          6541  2            ! Display the event kind (access, exception, instruction).
;  6428          6542  2            !
;  6429          6543  2            SELECTONE .EVENT_ENTRY [EVENT$B_CMD_KIND] OF
;  6430          6544  2                SET
;  6431          6545  2
;  6432          6546  2
;  6433          6547  2                ! This is an access (read, write, modify, execute) kind
;  6434          6548  2                ! of event point.  Display the address.
;  6435          6549  2                !
;  6436          6550  2                [EVENT$K_KIND_ACC]:
;  6437          6551  2                    BEGIN
;  6438          6552  2
;  6439          6553  2
;  6440          6554  2                    ! If this is not an /EXECUTE type (we're done with those), continue
;  6441          6555  2                    ! processing....
;  6442          6556  2                    !
;  6443          6557  2                    IF .EVENT_ENTRY [EVENT$B_SUB_KIND] NEQU EVENT$K_ACC_EXEC
;  6444          6558  2                    THEN
;  6445          6559  3                        BEGIN
;  6446          6560  3
;  6447          6561  3
;  6448          6562  3                        ! Print the access kind if read, write, modify, or return.
;  6449          6563  3                        ! If the address is an entry point, indicate that it is a
;  6450          6564  3                        ! routine.
;  6451          6565  3                        !
;  6452          6566  3                        DBG$PRINT   ($AC        (' !AC!AC'),
;  6453          6567  3                                    IF .EVENT_ENTRY [EVENT$B_CMD_TYPE] NEQU
;  6454          6568  3                                       EVENT$K_TYPE_WATCH
;  6455          6569  3                                    THEN
;  6456          6570  3                                       SELECTONE .EVENT_ENTRY [EVENT$B_SUB_KIND] OF
;  6457          6571  3                                       SET
;  6458          6572  3                                           [EVENT$K_ACC_READ]:
;  6459          6573  3                                               $AC ('on read of ');
;  6460          6574  3                                           [EVENT$K_ACC_WRIT]:
;  6461          6575  3                                               $AC ('on write of ');
;  6462          6576  3                                           [EVENT$K_ACC_MDFY]:
;  6463          6577  3                                               $AC ('on modify of ');
;  6464          6578  3                                           [EVENT$K_ACC_RTRN]:
;  6465          6579  3                                               $AC ('on return from ');
;  6466          6580  3                                       TES
;  6467          6581  3                                    ELSE
;  6468          6582  3                                       $AC ('of '),
;  6469          6583  3
;  6470          6584  3                                    IF .EVENT_ENTRY [EVENT$V_BREAK_ADDRESS]
;  6471          6585  3                                    THEN
;  6472          6586  3                                       $AC ('routine ')
;  6473          6587  3                                    ELSE
;  6474          6588  4                                       BEGIN
;  6475          6589  4                                       IF DBG$IS_IT_ENTRY
```

```
;  6476        6590  5                        (.EVENT_ENTRY [EVENT$L_ADDRESS] - 2)
;  6477        6591  5                    THEN
;  6478        6592  5                        $AC ('routine ')
;  6479        6593  5                    ELSE
;  6480        6594  5                        $AC ('')
;  6481        6595  5                    END
;  6482        6596  4                );
;  6483        6597  4
;  6484        6598  4
;  6485        6599  4                ! Print the address.
;  6486        6600  4                !
;  6487        6601  4                IF .EVENT_ENTRY [EVENT$B_CMD_TYPE] EQLU EVENT$K_TYPE_STEPS
;  6488        6602  4                THEN
;  6489        6603  5                    BEGIN
;  6490        6604  5                    LOCAL
;  6491        6605  5                        FROM_ADDR;
;  6492        6606  5
;  6493        6607  5
;  6494        6608  5                    ! Display 'ROUTINE' if the saved 'from' address points
;  6495        6609  5                    ! after an entry point.
;  6496        6610  5                    !
;  6497        6611  5                    FROM_ADDR = .EVENT_ENTRY [EVENT$L_USERS_PC];
;  6498        6612  5                    IF DBG$IS_IT_ENTRY (.FROM_ADDR - 2)
;  6499        6613  5                    THEN
;  6500        6614  6                        BEGIN
;  6501        6615  6                        DBG$PRINT ($AC ('routine '));
;  6502        6616  6                        FROM_ADDR = .FROM_ADDR - 2;
;  6503        6617  5                        END;
;  6504        6618  5
;  6505        6619  5
;  6506        6620  5                    ! Print the address, but symbolize it first.
;  6507        6621  5                    !
;  6508        6622  5                    DBG$PRINT_IDENTIFIER_PC (.FROM_ADDR);
;  6509        6623  5                    END
;  6510        6624  5
;  6511        6625  4                ELSE
;  6512        6626  4                    DBG$PRINT_IDENTIFIER (.EVENT_ENTRY [EVENT$L_PRIMARY]);
;  6513        6627  4
;  6514        6628  4                    END;
;  6515        6629  4
;  6516        6630  4
;  6517        6631  4                ! Depending on the command type....
;  6518        6632  4                !
;  6519        6633  4                IF .EVENT_ENTRY [EVENT$B_CMD_TYPE] EQLU EVENT$K_TYPE_STEPS
;  6520        6634  4                THEN
;  6521        6635  4                    DBG$PRINT ($AC (' to '))
;  6522        6636  4
;  6523        6637  4                ELSE
;  6524        6638  4                    DBG$PRINT ($AC (' at '));
;  6525        6639  4
;  6526        6640  4                END;
;  6527        6641  4
;  6528        6642  4
;  6529        6643  4            ! This is an instruction type.
;  6530        6644  4            !
;  6531        6645  4            [EVENT$K_KIND_INS]:
;  6532        6646  3                BEGIN
```

```
;  6533          6647  3
;  6534          6648  3
;  6535          6649  3                    ! Display the instruction type (calls, branches, all, or user
;  6536          6650  3                    ! list.
;  6537          6651  3                    !
;  6538          6652  3                    IF .EVENT_ENTRY [EVENT$B_CMD_TYPE] NEQU EVENT$K_TYPE_STEPS
;  6539          6653  3                    THEN
;  6540          6654  3                        DBG$PRINT
;  6541          6655  3                                ($AC    (' on !AC'),
;  6542          6656  3                                 SELECTONE .EVENT_ENTRY [EVENT$B_SUB_KIND] OF
;  6543          6657  3                                    SET
;  6544          6658  3                                    [EVENT$K_INS_CALL]:
;  6545          6659  3                                        $AC ('calls');
;  6546          6660  3                                    [EVENT$K_INS_BRAN]:
;  6547          6661  3                                        $AC ('branches');
;  6548          6662  3                                    [EVENT$K_INS_EVRY]:
;  6549          6663  3                                        $AC ('instruction');
;  6550          6664  3                                    [EVENT$K_INS_USER]:
;  6551          6665  3                                        $AC ('instruction(s)');
;  6552          6666  3                                    [EVENT$K_INS_LINE]:
;  6553          6667  3                                        $AC ('lines');
;  6554          6668  3                                    TES
;  6555          6669  3                                );
;  6556          6670  3
;  6557          6671  3
;  6558          6672  3                    ! Depending on the command type....
;  6559          6673  3                    !
;  6560          6674  3                    IF .EVENT_ENTRY [EVENT$B_CMD_TYPE] EQLU EVENT$K_TYPE_STEPS
;  6561          6675  3                    THEN
;  6562          6676  3                        DBG$PRINT ($AC (' to '))
;  6563          6677  3                    ELSE
;  6564          6678  3                        DBG$PRINT ($AC (' at '));
;  6565          6679  3                    END;
;  6566          6680  3
;  6567          6681  3
;  6568          6682  3                ! This is an exception type.
;  6569          6683  3                !
;  6570          6684  3                [EVENT$K_KIND_EXC]:
;  6571          6685  3                    BEGIN
;  6572          6686  3                    LOCAL
;  6573          6687  3                        DST_PTR : REF DST$RECORD,
;  6574          6688  3                        RST_PTR : REF RST$ENTRY,
;  6575          6689  3                        TRAP_OR_FAULT,
;  6576          6690  3                        LINE_NO,
;  6577          6691  3                        STMT_NO,
;  6578          6692  3                        STATUS;
;  6579          6693  3
;  6580          6694  3
;  6581          6695  3                    ! Determine the nature (trap or fault) of the exception.
;  6582          6696  3                    !
;  6583          6697  3                    IF .EXCEPTION_TYPE [STS$V_FAC_NO] EQL 0
;  6584          6698  4                    THEN
;  6585          6699  4                        BEGIN
;  6586          6700  4                        SELECTONE .EXCEPTION_TYPE OF
;  6587          6701  4                            SET
;  6588          6702  4
;  6589          6703  4                            [SS$_ASTFLT, SS$_BREAK, SS$_COMPAT,
```

```
;  6590          6704  4                    SS$_OPCCUS, SS$_OPCDEC, SS$_RADRMOD, SS$_ROPRAND]:
;  6591          6705  4                        TRAP_OR_FAULT = FAULT_EXC;
;  6592          6706  4
;  6593          6707  4                    [OTHERWISE]:
;  6594          6708  4                        TRAP_OR_FAULT = TRAP_EXC;
;  6595          6709  4
;  6596          6710  4                    TES;
;  6597          6711  4
;  6598          6712  4                END
;  6599          6713  4
;  6600          6714  4
;  6601          6715  4            ! If the facility is not SS$, then we have a trap, meaning that the
;  6602          6716  4            ! PC points to the instruction after the LIB$SIGNAL or LIB$STOP call
;  6603          6717  4            ! that raised the exception.  (Non-SS$ exceptions can only be sig-
;  6604          6718  4            ! nalled, not raised by the hardware.)
;  6605          6719  4            !
;  6606          6720  3            ELSE
;  6607          6721  3                TRAP_OR_FAULT = TRAP_EXC;
;  6608          6722  3
;  6609          6723  3
;  6610          6724  3            ! The PC is the address to display.
;  6611          6725  3            !
;  6612          6726  3            ADDRESS = .DBG$RUNFRAME [DBG$L_USER_PC];
;  6613          6727  3
;  6614          6728  3
;  6615          6729  3            ! Display the appropriate message.
;  6616          6730  3            !
;  6617          6731  3            DBG$PRINT ($AC (' on exception '));
;  6618          6732  3            IF .TRAP_OR_FAULT EQL TRAP_EXC
;  6619          6733  3            THEN
;  6620          6734  3                DBG$PRINT ($AC ('preceding '))
;  6621          6735  3
;  6622          6736  3                ELSE
;  6623          6737  3                    DBG$PRINT ($AC ('at '));
;  6624          6738  3
;  6625          6739  3                END;
;  6626          6740  3
;  6627          6741  3            TES;
;  6628          6742  3
;  6629          6743  3
;  6630          6744  2            ! If this is a /MODIFY event (we used ACCVIO's), the address of the
;  6631          6745  2            ! offending instruction is in ACCVIO_PC, otherwise it's just the
;  6632          6746  2            ! current PC.
;  6633          6747  2            !
;  6634          6748  2            IF (.EVENT_ENTRY [EVENT$B_CMD_KIND] EQLU EVENT$K_KIND_ACC) AND
;  6635          6749  2               (.EVENT_ENTRY [EVENT$B_SUB_KIND] EQLU EVENT$K_ACC_MDFY)
;  6636          6750  2            THEN
;  6637          6751  2                ADDRESS = .ACCVIO_PC
;  6638          6752  2
;  6639          6753  2            ELSE
;  6640          6754  2                ADDRESS = .DBG$RUNFRAME [DBG$L_USER_PC];
;  6641          6755  2
;  6642          6756  2
;  6643          6757  2            ! Symbolize and print address (and instruction)
;  6644          6758  2            !
;  6645          6759  2            PRINT_PC_AND_INSTRUCTION (.ADDRESS, .EVENT_ENTRY);
;  6646          6760  2            DBG$NEWLINE ();
```

```
;  6647        6761  2
;  6648        6762
;  6649        6763          ! If this is a modify-type, display the old and new values, and the
;  6650        6764          ! BREAK message.
;  6651        6765          !
;  6652        6766          IF .EVENT_ENTRY [EVENT$B_SUB_KIND] EQLU EVENT$K_ACC_MDFY
;  6653        6767          THEN
;  6654        6768              BEGIN
;  6655        6769
;  6656        6770
;  6657        6771              ! Print the source line where the actual variable-access occurred if
;  6658        6772              ! appropriate.
;  6659        6773              !
;  6660        6774              IF (.EVENT_ENTRY[EVENT$V_OVRD_SOURCE] AND
;  6661        6775                                  .EVENT_ENTRY[EVENT$V_STEP_SOURCE]) OR
;  6662        6776                 ((NOT .EVENT_ENTRY[EVENT$V_OVRD_SOURCE]) AND
;  6663        6777                                  .DBG$GB_STP_PTR[EVENT$V_STEPPING_SOURCE])
;  6664        6778              THEN
;  6665        6779                  PRINT_SOURCE(.ADDRESS);
;  6666        6780
;  6667        6781
;  6668        6782              ! Print the old value and the new value of the watched location.
;  6669        6783              !
;  6670        6784              DBG$PRINT($AC('   old value: '));
;  6671        6785              DBG$PRINT_VALUE(.OLDVALUE, .DBG$GB_RADIX[DBG$B_RADIX_OUTPUT_OVER],
;  6672        6786                                                  .DBG$G[_SIGN_FLAG, FALSE);
;  6673        6787              DBG$NEWLINE();
;  6674        6788              DBG$PRINT($AC('   new value: '));
;  6675        6789              DBG$PRINT_VALUE(.NEWVALUE, .DBG$GB_RADIX[DBG$B_RADIX_OUTPUT_OVER],
;  6676        6790                                                  .DBG$G[_SIGN_FLAG, FALSE);
;  6677        6791              DBG$NEWLINE();
;  6678        6792
;  6679        6793
;  6680        6794              ! Display the 'BREAK AT <address>' message.
;  6681        6795              !
;  6682        6796              DBG$PRINT (IF .EVENT_ENTRY [EVENT$B_CMD_TYPE] NEQU
;  6683        6797                            EVENT$K_TYPE_TRACE
;  6684        6798                         THEN
;  6685        6799                             $AC ('break at ')
;  6686        6800                         ELSE
;  6687        6801                             $AC ('trace at ')
;  6688        6802                        );
;  6689        6803
;  6690        6804              ! Symbolize and print address (and instruction)
;  6691        6805              !
;  6692        6806              PRINT_PC_AND_INSTRUCTION (.DBG$RUNFRAME [DBG$L_USER_PC], .EVENT_ENTRY);
;  6693        6807              DBG$NEWLINE ();
;  6694        6808              END;
;  6695        6809
;  6696        6810
;  6697        6811          ! Display the source line, if source stepping is indicated.
;  6698        6812          !
;  6699        6813          IF .TYPE_SOURCE AND
;  6700        6814             ((.EVENT_ENTRY[EVENT$V_OVRD_SOURCE] AND
;  6701        6815                                  .EVENT_ENTRY[EVENT$V_STEP_SOURCE]) OR
;  6702        6816              ((NOT .EVENT_ENTRY[EVENT$V_OVRD_SOURCE]) AND
;  6703        6817                                  .DBG$GB_STP_PTR[EVENT$V_STEPPING_SOURCE]))
```

```
;  6704          6818  2     THEN
;  6705          6819  2         BEGIN
;  6706          6820  2         PRINT_SOU      BG$RUNFRAME [DBG$L_USER_PC]);
;  6707          6821  2         TYPE_SOURCE    SE;
;  6708          6822  2         END;
;  6709          6823  2
;  6710          6824  2
;  6711          6825  2     ! We are all e    return.
;  6712          6826  2     !
;  6713          6827  2     RETURN;
;  6714          6828  2
;  6715          6829  1     END;


                                                                .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,O

                                                  43 41 21 03   00279 P.ACH:  .ASCII  <3>\!AC\                          ;
                                               68 61 65 72 62   0027D P.ACI:  .ASCII  <5>\break\                        ;
                                               65 63 61 72 74   00283 P.ACJ:  .ASCII  <5>\trace\                        ;
                                               68 63 74 61 77   00289 P.ACK:  .ASCII  <5>\watch\                        ;
                                            64 65 70 70 65 74   0028F P.ACL:  .ASCII  <7>\stepped\                       ;
                                            43 41 21 43 41 21   00297 P.ACM:  .ASCII  <7>\ !AC!AC\                       ;
                                  20 66 6F 20 64 61 65 72 20   0029F P.ACN:  .ASCII  <11>\on read of \                   ;
                               20 66 6F 20 65 74 69 72 77 20   002AB P.ACO:  .ASCII  <12>\on write of \                  ;
                            20 66 6F 20 79 66 69 64 6F 6D 20   002B8 P.ACP:  .ASCII  <13>\on modify of \               ;
6D 6F 72 66 20 6E 72 75 74 65 72 20 6E 6F                   002C6 P.ACQ:  .ASCII  <15>\on return from \              ;
                                                        20  002D5
                                                  20 66 6F   002D6 P.ACR:  .ASCII  <3>\of \                           ;
                                  20 65 6E 69 74 75 6F 72   002DA P.ACS:  .ASCII  <8>\routine \                        ;
                                  20 65 6E 69 74 75 6F 72   002E3 P.ACT:  .ASCII  <8>\routine \                        ;
                                                        00  002EC P.ACU:  .ASCII  <0>                                  ;
                                  20 65 6E 69 74 75 6F 72   002ED P.ACV:  .ASCII  <8>\routine \                        ;
                                                  20 6F 74   002F6 P.ACW:  .ASCII  <4>\ to \                          ;
                                                  20 74 61   002FB P.ACX:  .ASCII  <4>\ at \                          ;
                                           43 41 21 20 6E 6F   00300 P.ACY:  .ASCII  <7>\ on !AC\                      ;
                                                  73 6C 6C   00308 P.ACZ:  .ASCII  <5>\calls\                         ;
                                  73 65 68 63 6E 61 72 62   0030E P.ADA:  .ASCII  <8>\branches\                       ;
                               6E 6F 69 74 63 75 72 74 73 6E   00317 P.ADB:  .ASCII  <11>\instruction\                 ;
29 73 28 6E 6F 69 74 63 75 72 74 73 6E 69 0E                00323 P.ADC:  .ASCII  <14>\instruction(s)\                ;
                                                  73 65 6E   00332 P.ADD:  .ASCII  <5>\lines\                         ;
                                                  20 6F 74   00338 P.ADE:  .ASCII  <4>\ to \                          ;
                                                  20 74 61   0033D P.ADF:  .ASCII  <4>\ at \                          ;
20 6E 6F 69 74 70 65 63 78 65 20 6E 6F 20 0E                00342 P.ADG:  .ASCII  <14>\ on exception \                ;
                                  20 67 6E 69 64 65 63 65   00351 P.ADH:  .ASCII  <10>\preceding \                    ;
                                                  20 74 61   0035C P.ADI:  .ASCII  <3>\at \                           ;
20 3A 65 75 6C 61 76 20 64 6C 6F 20 20 20 0E                00360 P.ADJ:  .ASCII  <14>\   old value: \                ;
20 3A 65 75 6C 61 76 20 77 65 6E 20 20 20 0E                0036F P.ADK:  .ASCII  <14>\   new value: \                ;
                               20 74 61 20 68 61 65 72 62   0037E P.ADL:  .ASCII  <9>\break at \                     ;
                               20 74 61 20 65 63 61 72 74   00388 P.ADM:  .ASCII  <9>\trace at \                     ;


                                                                .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,O

                                                  OFFC 00000 ANNOUNCE_EVENT:
                                                             .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11          ; 6476
```

```
                5B 00000000G   00  9E 00002          MOVAB   DBG$IS_IT_ENTRY, R11                    ;
                5A 00000000G   00  9E 00009          MOVAB   DBG$NEQLINE, R10                        ;
                59 00000000G   00  9E 00010          MOVAB   DBG$RUNFRAME+64, R9                     ;
                58 00000000'   EF  9E 00017          MOVAB   TYPE_SOURCE, R8                         ;
                57 00000000G   00  9E 0001E          MOVAB   DBG$PRINT, R7                           ;
                56 00000000'   EF  9E 00025          MOVAB   P.ACI, R6                               ;
                5E       FF68  CE  9E 0002C          MOVAB   -152(SP), SP                            ;
                52         04  AC  D0 00031          MOVL    EVENT_ENTRY, R2                         ; 6508
                54         14  A2  9E 00035          MOVAB   20(R2), R4                              ;
                01         03  A4  E9 00039          BLBC    3(R4), 1$                               ;
                                   04 0003D          RET                                             ;
00000000G       00         80  8F  88 0003E 1$:      BISB2   #128, DBG$GL_OUTPRAB+7                  ; 6513
          EC    AD 010E0000  8F  D0 00046          MOVL    #17694720, STRING_DESCRIPTOR            ; 6521
                         F0  AD  D4 0004E          CLRL    STRING_DESCRIPTOR+4                     ; 6520
                             64  95 00051          TSTB    (R4)                                    ; 6529
                             05  12 00053          BNEQ    2$                                      ;
                50              66  9E 00055          MOVAB   P.ACI, R0                           ; 6530
                                24  11 00058          BRB     6$                                  ;
                01              64  91 0005A 2$:      CMPB    (R4), #1                            ; 6531
                                06  12 0005D          BNEQ    3$                                  ;
                50         06   A6  9E 0005F          MOVAB   P.ACJ, R0                           ; 6532
                                19  11 00063          BRB     6$                                  ;
                02              64  91 00065 3$:      CMPB    (R4), #2                            ; 6533
                                06  12 00068          BNEQ    4$                                  ;
                50         0C   A6  9E 0006A          MOVAB   P.ACK, R0                           ; 6534
                                0E  11 0006E          BRB     6$                                  ;
                03              64  91 00070 4$:      CMPB    (R4), #3                            ; 6535
                                05  13 00073          BEQL    5$                                  ;
                7E              01  CE 00075          MNEGL   #1, -(SP)                           ;
                                06  11 00078          BRB     7$                                  ;
                50         12   A6  9E 0007A 5$:      MOVAB   P.ACL, R0                           ; 6536
                                50  DD 0007E 6$:      PUSHL   R0                                  ;
                         FC     A6  9F 00080 7$:      PUSHAB  P.ACH                               ; 6526
                67              02  FB 00083          CALLS   #2, DBG$PRINT                       ;
                01              A4  95 00086          TSTB    1(R4)                               ; 6550
                                03  13 00089          BEQL    8$                                  ;
                              00AF  31 0008B          BRW     23$                                 ;
                03         02   A4  91 0008E 8$:      CMPB    2(R4), #3                           ; 6557
                                03  12 00092          BNEQ    9$                                  ;
                              0097  31 00094          BRW     21$                                 ;
          06    19         A2      05  E1 00097 9$:      BBC     #5, 25(R2), 10$                 ; 6584
                50         5D   A6  9E 0009C          MOVAB   P.ACS, R0                           ; 6586
                                15  11 000A0          BRB     12$                                 ;
          7E    2C         A2      02  C3 000A2 10$:     SUBL3   #2, 44(R2), -(SP)               ; 6590
                6B              01  FB 000A7          CALLS   #1, DBG$IS_IT_ENTRY                 ;
                06              50  E9 000AA          BLBC    R0, 11$                             ;
                50         66   A6  9E 000AD          MOVAB   P.ACT, R0                           ; 6592
                                04  11 000B1          BRB     12$                                 ;
                50         6F   A6  9E 000B3 11$:     MOVAB   P.ACU, R0                           ; 6594
                                50  DD 000B7 12$:     PUSHL   R0                                  ; 6588
                02              64  91 000B9          CMPB    (R4), #2                            ; 6567
                                34  13 000BC          BEQL    17$                                 ;
                02         A4   95 000BE             TSTB    2(R4)                               ; 6572
                                06  12 000C1          BNEQ    13$                                 ;
                50         22   A6  9E 000C3          MOVAB   P.ACN, R0                           ; 6573
                                2D  11 000C7          BRB     18$                                 ;
                01         02   A4  91 000C9 13$:     CMPB    2(R4), #1                           ; 6574
```

```
                              06  12 000CD        BNEQ    14$                                   ;
                  50      2E  A6  9E 000CF        MOVAB   P.ACO, R0                             ; 6575
                              21  11 000D3        BRB     18$                                   ;
                  02      02  A4  91 000D5  14$:  CMPB    2(R4), #2                             ; 6576
                              06  12 000D9        BNEQ    15$                                   ;
                  50      3B  A6  9E 000DB        MOVAB   P.ACP, R0                             ; 6577
                              15  11 000DF        BRB     18$                                   ;
                  04      02  A4  91 000E1  15$:  CMPB    2(R4), #4                             ; 6578
                              05  13 000E5        BEQL    16$                                   ;
                  50          01  CE 000E7        MNEGL   #1, R0                                ;
                              0A  11 000EA        BRB     18$                                   ;
                  50      49  A6  9E 000EC  16$:  MOVAB   P.ACQ, R0                             ; 6579
                              04  11 000F0        BRB     18$                                   ; 6570
                  50      59  A6  9E 000F2  17$:  MOVAB   P.ACR, R0                             ; 6582
                              50  DD 000F6  18$:  PUSHL   R0                                    ;
                          1A  A6  9F 000F8        PUSHAB  P.ACM                                 ; 6566
                  67          03  FB 000FB        CALLS   #3, DBG$PRINT                         ;
                  03          64  91 000FE        CMPB    (R4), #3                              ; 6601
                              21  12 00101        BNEQ    20$                                   ;
                  53      20  A2  D0 00103        MOVL    32(R2), FROM_ADDR                     ; 6611
                          FE  A3  9F 00107        PUSHAB  -2(FROM_ADDR)                         ; 6612
                  6B          01  FB 0010A        CALLS   #1, DBG$IS_IT_ENTRY                   ;
                  09          50  E9 0010D        BLBC    R0, 19$                               ;
                          70  A6  9F 00110        PUSHAB  P.ACV                                 ; 6615
                  67          01  FB 00113        CALLS   #1, DBG$PRINT                         ;
                  53          02  C2 00116        SUBL2   #2, FROM_ADDR                         ; 6616
                              53  DD 00119  19$:  PUSHL   FROM_ADDR                             ; 6622
       00000000G  00          01  FB 0011B        CALLS   #1, DBG$PRINT_IDENTIFIER_PC           ;
                              0A  11 00122        BRB     21$                                   ; 6601
                          28  A2  DD 00124  20$:  PUSHL   40(R2)                                ; 6626
       00000000G  00          01  FB 00127        CALLS   #1, DBG$PRINT_IDENTIFIER              ;
                  03          64  91 0012E  21$:  CMPB    (R4), #3                              ; 6633
                              05  12 00131        BNEQ    22$                                   ;
                          79  A6  9F 00133        PUSHAB  P.ACW                                 ; 6635
                              6C  11 00136        BRB     33$                                   ;
                          7E  A6  9F 00138  22$:  PUSHAB  P.ACX                                 ; 6638
                              67  11 0013B        BRB     33$                                   ;
                  01      01  A4  91 0013D  23$:  CMPB    1(R4), #1                             ; 6645
                              63  12 00141        BNEQ    34$                                   ;
                  03          64  91 00143        CMPB    (R4), #3                              ; 6652
                              4D  13 00146        BEQL    31$                                   ;
                  05      02  A4  91 00148        CMPB    2(R4), #5                             ; 6658
                              07  12 0014C        BNEQ    24$                                   ;
                  50    008B  C6  9E 0014E        MOVAB   P.ACZ, R0                             ; 6659
                              37  11 00153        BRB     29$                                   ;
                  06      02  A4  91 00155  24$:  CMPB    2(R4), #6                             ; 6660
                              07  12 00159        BNEQ    25$                                   ;
                  50    0091  C6  9E 0015B        MOVAB   P.ADA, R0                             ; 6661
                              2A  11 00160        BRB     29$                                   ;
                  08      02  A4  91 00162  25$:  CMPB    2(R4), #8                             ; 6662
                              07  12 00166        BNEQ    26$                                   ;
                  50    009A  C6  9E 00168        MOVAB   P.ADB, R0                             ; 6663
                              1D  11 0016D        BRB     29$                                   ;
                  09      02  A4  91 0016F  26$:  CMPB    2(R4), #9                             ; 6664
                              07  12 00173        BNEQ    27$                                   ;
                  50    00A6  C6  9E 00175        MOVAB   P.ADC, R0                             ; 6665
                              10  11 0017A        BRB     29$                                   ;
```

```
                07      02      A4  91 0017C 27$:    CMPB    2(R4), #7                       ; 6666
                                05  13 00180         BEQL    28$
                7E                  01  CE 00182         MNEGL   #1, -(SP)
                                07  11 00185         BRB     30$
                50      00B5    C6  9E 00187 28$:    MOVAB   P.ADD, R0                       ; 6667
                                50  DD 0018C 29$:    PUSHL   R0
                        0083    C6  9F 0018E 30$:    PUSHAB  P.ACY                           ; 6655
                67              02  FB 00192         CALLS   #2, DBG$PRINT
                03              64  91 00195 31$:    CMPB    (R4), #3                        ; 6674
                                06  12 00198         BNEQ    32$
                        00BB    C6  9F 0019A         PUSHAB  P.ADE                           ; 6676
                                78  11 0019E         BRB     39$
                        00C0    C6  9F 001A0 32$:    PUSHAB  P.ADF                           ; 6678
                                72  11 001A4 33$:    BRB     39$
                02      01      A4  91 001A6 34$:    CMPB    1(R4), #2                       ; 6684
                                6F  12 001AA         BNEQ    40$
            0FFF    8F      0A  AC  B3 001AC         BITW    EXCEPTION_TYPE+2, #4095         ; 6697
                                48  12 001B2         BNEQ    36$
                50      08      AC  D0 001B4         MOVL    EXCEPTION_TYPE, R0              ; 6700
        0000040C    8F              50  D1 001B8         CMPL    R0, #1036                       ; 6703
                                36  13 001BF         BEQL    35$
        00000414    8F              50  D1 001C1         CMPL    R0, #1044
                                2D  13 001C8         BEQL    35$
        0000042C    8F              50  D1 001CA         CMPL    R0, #1068
                                24  13 001D1         BEQL    35$
        00000434    8F              50  D1 001D3         CMPL    R0, #1076
                                1B  13 001DA         BEQL    35$
        0000043C    8F              50  D1 001DC         CMPL    R0, #1084
                                12  13 001E3         BEQL    35$
        0000044C    8F              50  D1 001E5         CMPL    R0, #1100
                                09  13 001EC         BEQL    35$
        00000454    8F              50  D1 001EE         CMPL    R0, #1108
                                05  12 001F5         BNEQ    36$
                53                  02  D0 001F7 35$:    MOVL    #2, TRAP_OR_FAULT               ; 6705
                                03  11 001FA         BRB     37$
                53                  01  D0 001FC 36$:    MOVL    #1, TRAP_OR_FAULT               ; 6721
                55                  69  D0 001FF 37$:    MOVL    DBG$RUNFRAME+64, ADDRESS        ; 6726
                        00C5    C6  9F 00202         PUSHAB  P.ADG                           ; 6731
                67              01  FB 00206         CALLS   #1, DBG$PRINT
                01                  53  D1 00209         CMPL    TRAP_OR_FAULT, #1               ; 6732
                                06  12 0020C         BNEQ    38$
                        00D4    C6  9F 0020E         PUSHAB  P.ADH                           ; 6734
                                04  11 00212         BRB     39$
                        00DF    C6  9F 00214 38$:    PUSHAB  P.ADI                           ; 6737
                67              01  FB 00218 39$:    CALLS   #1, DBG$PRINT
                        01      A4  95 0021B 40$:    TSTB    1(R4)                           ; 6748
                                0C  12 0021E         BNEQ    41$
                02      02      A4  91 00220         CMPB    2(R4), #2                       ; 6749
                                06  12 00224         BNEQ    41$
                55      04      A8  D0 00226         MOVL    ACCVIO_PC, ADDRESS              ; 6751
                                03  11 0022A         BRB     42$
                55                  69  D0 0022C 41$:    MOVL    DBG$RUNFRAME+64, ADDRESS        ; 6754
                                52  DD 0022F 42$:    PUSHL   R2                              ; 6759
                                55  DD 00231         PUSHL   ADDRESS
            0000V   CF              02  FB 00233         CALLS   #2, PRINT_PC_AND_INSTRUCTION
                6A                  00  FB 00238         CALLS   #0, DBG$NEWLINE             ; 6760
                02      02      A4  91 0023B         CMPB    2(R4), #2                       ; 6766
```

```
                                   03 13 0023F         BEQL    43$                                  ;
                                 0089 31 00241         BRW     49$                                  ;
 0A      18  A2                    04 E1 00244 43$:    BBC     #4, 24(R2), 44$                      ; 6774
 10      18  A2                    05 E0 00249         BBS     #5, 24(R2), 45$                      ; 6775
 12      18  A2                    04 E0 0024E         BBS     #4, 24(R2), 46$                      ; 6776
             50 00000000G          00 D0 00253 44$:    MOVL    DBG$GB_STP_PTR, R0                   ; 6777
 07          60                    0A E1 0025A         BBC     #10, (R0), 46$                       ;
                                   55 DD 0025E 45$:    PUSHL   ADDRESS                              ; 6779
       0000V CF                    01 FB 00260         CALLS   #1, PRINT_SOURCE                     ;
                        00E3       C6 9F 00265 46$:    PUSHAB  P.ADJ                                ; 6784
             67                    01 FB 00269         CALLS   #1, DBG$PRINT                        ;
                                   7E D4 0026C         CLRL    -(SP)                                ; 6785
                00000000G          00 DD 0026E         PUSHL   DBG$GL_SIGN_FLAG                     ; 6786
             7E 00000000G          00 9A 00274         MOVZBL  DBG$GB_RADIX+2, -(SP)                ; 6785
                          FC       A8 DD 0027B         PUSHL   OLDVALUE                             ;
   00000000G 00                    04 FB 0027E         CALLS   #4, DBG$PRINT_VALUE                  ;
             6A                    00 FB 00285         CALLS   #0, DBG$NEWLINE                      ; 6787
                        00F2       C6 9F 00288         PUSHAB  P.ADK                                ; 6788
             67                    01 FB 0028C         CALLS   #1, DBG$PRINT                        ;
                                   7E D4 0028F         CLRL    -(SP)                                ; 6789
                00000000G          00 DD 00291         PUSHL   DBG$GL_SIGN_FLAG                     ; 6790
             7E 00000000G          00 9A 00297         MOVZBL  DBG$GB_RADIX+2, -(SP)                ; 6789
                          F8       A8 DD 0029E         PUSHL   NEWVALUE                             ;
   00000000G 00                    04 FB 002A1         CALLS   #4, DBG$PRINT_VALUE                  ;
             6A                    00 FB 002A8         CALLS   #0, DBG$NEWLINE                      ; 6791
             01                    64 91 002AB         CMPB    (R4), #1                             ; 6796
                                   07 13 002AE         BEQL    47$                                  ;
             50         0101       C6 9E 002B0         MOVAB   P.ADL, R0                            ; 6799
                                   05 11 002B5         BRB     48$                                  ;
             50         010B       C6 9E 002B7 47$:    MOVAB   P.ADM, R0                            ; 6801
                                   50 DD 002BC 48$:    PUSHL   R0                                   ;
             67                    01 FB 002BE         CALLS   #1, DBG$PRINT                        ; 6796
                                   52 DD 002C1         PUSHL   R2                                   ; 6806
                                   69 DD 002C3         PUSHL   DBG$RUNFRAME+64                      ;
       0000V CF                    02 FB 002C5         CALLS   #2, PRINT_PC_AND_INSTRUCTION         ;
             6A                    00 FB 002CA         CALLS   #0, DBG$NEWLINE                      ; 6807
             23                    68 E9 002CD 49$:    BLBC    TYPE_SOURCE, 52$                     ; 6813
 0A      18  A2                    04 E1 002D0         BBC     #4, 24(R2), 50$                      ; 6814
 10      18  A2                    05 E0 002D5         BBS     #5, 24(R2), 51$                      ; 6815
 14      18  A2                    04 E0 002DA         BBS     #4, 24(R2), 52$                      ; 6816
             50 00000000G          00 D0 002DF 50$:    MOVL    DBG$GB_STP_PTR, R0                   ; 6817
 09          60                    0A E1 002E6         BBC     #10, (R0), 52$                       ;
                                   69 DD 002EA 51$:    PUSHL   DBG$RUNFRAME+64                      ; 6820
       0000V CF                    01 FB 002EC         CALLS   #1, PRINT_SOURCE                     ;
                                   68 D4 002F1         CLRL    TYPE_SOURCE                          ; 6821
                                      04 002F3 52$:    RET                                          ; 6829

; Routine Size:  756 bytes,    Routine Base:  DBG$CODE + 25EA
```

```
;  6717          6830  1 ROUTINE INSQUE1(ENTRY_ADDRESS, QUEUE_ADDRESS): NOVALUE =
;  6718          6831  1 !
;  6719          6832  1 ! FUNCTION
;  6720          6833  1 !       This routine adds an entry into a queue.  This routine is used when
;  6721          6834  1 !       the forward and backward links are not the 0th and 1st longwords in
;  6722          6835  1 !       the queue head and queue entry, but the 2nd and 3rd.
;  6723          6836  1 !
;  6724          6837  1 ! INPUTS
;  6725          6838  1 !       QUEUE_ADDRESS :         Pointer to queue head.
;  6726          6839  1 !       ENTRY_ADDRESS :         Pointer to entry.
;  6727          6840  1 !
;  6728          6841  1 ! OUTPUTS
;  6729          6842  1 !       None
;  6730          6843  1 !
;  6731          6844  2     BEGIN
;  6732          6845  2
;  6733          6846  2     MAP
;  6734          6847  2         ENTRY_ADDRESS : REF QUEUE_HEAD,
;  6735          6848  2         QUEUE_ADDRESS : REF QUEUE_HEAD;
;  6736          6849  2
;  6737          6850  2     LOCAL
;  6738          6851  2         NEXTS_ADDRESS : REF QUEUE_HEAD;     ! Next entry's pointer.
;  6739          6852  2
;  6740          6853  2
;  6741          6854  2     ! Point to the current 1st (next) entry.
;  6742          6855  2     !
;  6743          6856  2     NEXTS_ADDRESS = .QUEUE_ADDRESS [L_QUEUE_FLINK1];
;  6744          6857  2
;  6745          6858  2
;  6746          6859  2     ! Link the queue entry into the queue at the head.
;  6747          6860  2     !
;  6748          6861  2     ENTRY_ADDRESS [L_QUEUE_FLINK1] = .QUEUE_ADDRESS [L_QUEUE_FLINK1];
;  6749          6862  2     ENTRY_ADDRESS [L_QUEUE_BLINK1] = .QUEUE_ADDRESS;
;  6750          6863  2
;  6751          6864  2
;  6752          6865  2     ! Link the queue to the queue entry.
;  6753          6866  2     !
;  6754          6867  2     QUEUE_ADDRESS [L_QUEUE_FLINK1] = .ENTRY_ADDRESS;
;  6755          6868  2     NEXTS_ADDRESS [L_QUEUE_BLINK1] = .ENTRY_ADDRESS;
;  6756          6869  1     END;


                                        0004 00000 INSQUE1:.WORD    Save R2                                 ;  6830
                    50      08      AC  D0 00002       MOVL     QUEUE_ADDRESS, R0                       ;  6856
                    52      08      A0  D0 00006       MOVL     8(R0), NEXTS_ADDRESS
                    51      04      AC  D0 0000A       MOVL     ENTRY_ADDRESS, R1                       ;  6861
                08  A1      08      A0  D0 0000E       MOVL     8(R0), 8(R1)
                0C  A1              50  D0 00013       MOVL     R0, 12(R1)                              ;  6862
                08  A0              51  D0 00017       MOVL     R1, 8(R0)                               ;  6867
                0C  A2              51  D0 0001B       MOVL     R1, 12(NEXTS_ADDRESS)                   ;  6868
                                        04 0001F       RET                                              ;  6869

; Routine Size:  32 bytes,    Routine Base:  DBG$CODE + 28DE
```

```
;  6758          6870  1 ROUTINE REMQUE1(ENTRY_ADDRESS), NOVALUE =
;  6759          6871  1 !
;  6760          6872  1 ! FUNCTION
;  6761          6873  1 !       This routine deletes an entry from a queue.  This routine is used when
;  6762          6874  1 !       the forward and backward links are not the 0th and 1st longwords in
;  6763          6875  1 !       the queue head and queue entry, but the 2nd and 3rd.
;  6764          6876  1 !
;  6765          6877  1 ! INPUTS
;  6766          6878  1 !       ENTRY_ADDRESS :         Pointer to entry.
;  6767          6879  1 !
;  6768          6880  1 ! OUTPUTS
;  6769          6881  1 !       None
;  6770          6882  1 !
;  6771          6883  2   BEGIN
;  6772                2
;  6773          6885  2   MAP
;  6774          6886  2       ENTRY_ADDRESS : REF QUEUE_HEAD;
;  6775          6887  2
;  6776          6888  2   LOCAL
;  6777          6889  2       NEXTS_ADDRESS : REF QUEUE_HEAD,
;  6778          6890  2       QUEUE_ADDRESS : REF QUEUE_HEAD;
;  6779          6891  2
;  6780          6892  2
;  6781          6893  2   ! Point to the queue head and next entry.
;  6782          6894  2   !
;  6783          6895  2   QUEUE_ADDRESS = .ENTRY_ADDRESS [L_QUEUE_BLINK1];
;  6784          6896  2   NEXTS_ADDRESS = .ENTRY_ADDRESS [L_QUEUE_FLINK1];
;  6785          6897  2
;  6786          6898  2
;  6787          6899  2   ! Unlink the entry from the queue.
;  6788          6900  2   !
;  6789          6901  2   QUEUE_ADDRESS [L_QUEUE_FLINK1] = .NEXTS_ADDRESS;
;  6790          6902  2   NEXTS_ADDRESS [L_QUEUE_BLINK1] = .QUEUE_ADDRESS;
;  6791          6903  1   END;


                                    0000 00000 REMQUE1:.WORD    Save nothing                                    ;  6870
                      50      04    AC   D0 00002          MOVL    ENTRY_ADDRESS, R0                           ;  6895
                      50      08    A0   7D 00006          MOVQ    8(R0), NEXTS_ADDRESS                        ;  6896
                08    A1            50   D0 0000A          MOVL    NEXTS_ADDRESS, 8(QUEUE_ADDRESS)             ;  6901
                0C    A0            51   D0 0000E          MOVL    QUEUE_ADDRESS, 12(NEXTS_ADDRESS)            ;  6902
                                          04 00012          RET                                             ;  6903

; Routine Size:  19 bytes,    Routine Base:  DBG$CODE + 28FE
```

```
;  6793          6904  1 ROUTINE PRINT_SOURCE(ADDRESS): NOVALUE =
;  6794          6905  2     BEGIN
;  6795          6906  2     BUILTIN
;  6796          6907  2         FP;
;  6797          6908  2
;  6798          6909  2     .FP = DBG$FINAL_HANDL;
;  6799          6910  2     DBG$SRC_TYPE_PC_SOURCE (.ADDRESS, .ADDRESS, 0, TRUE);
;  6800          6911  1     END;



                                        0000 00000 PRINT_SOURCE:
                                                           .WORD   Save nothing                                    ; 6904
                            6D 00000000G   00   9E 00002           MOVAB   DBG$FINAL_HANDL, (FP)                   ; 6909
                                           01   DD 00009           PUSHL   #1                                      ; 6910
                                           7E   D4 0000B           CLRL    -(SP)                                   ;
                                     04    AC   DD 0000D           PUSHL   ADDRESS                                 ;
                                     04    AC   DD 00010           PUSHL   ADDRESS                                 ;
                         00000000G   00         04   FB 00013     CALLS   #4, DBG$SRC_TYPE_PC_SOURCE              ;
                                                04 0001A           RET                                             ; 6911

; Routine Size:  27 bytes,    Routine Base:  DBG$CODE + 2911
```

```
;  6802     6912  1 ROUTINE PRINT_PC_AND_INSTRUCTION(ADDRESS, EVENT): NOVALUE =
;  6803     6913  2     BEGIN
;  6804     6914  2     MAP
;  6805     6915  2         ADDRESS :       REF VECTOR [, BYTE],
;  6806     6916  2         EVENT   :       REF EVENT$EVENT_DESCRIPTOR;
;  6807     6917  2
;  6808     6918  2     LOCAL ADDRESS_2;
;  6809     6919  2     BUILTIN
;  6810     6920  2         FP;
;  6811     6921  2
;  6812     6922  2
;  6813     6923  2     ! Establish a handler for errors. This is in case the instruction
;  6814     6924  2     ! we have stepped to is invalid, and the INS_DECODE routine signals
;  6815     6925  2     ! an error.
;  6816     6926  2     !
;  6817     6927  2     .FP = PRINT_PC_INST_HANDLER;
;  6818     6928  2
;  6819     6929  2
;  6820     6930  2     ! Display 'ROUTINE' if the current address points after an entry point.
;  6821     6931  2     !
;  6822     6932  2     IF .EVENT [EVENT$V_BREAK_ADDRESS]
;  6823     6933  2     THEN
;  6824     6934  2         DBG$PRINT ($AC ('routine '))
;  6825     6935  2
;  6826     6936  2     ELSE
;  6827     6937  2         BEGIN
;  6828     6938  2         IF DBG$IS_IT_ENTRY ((ADDRESS_2 = .ADDRESS - 2))
;  6829     6939  2         THEN
;  6830     6940  2             DBG$PRINT ($AC ('routine '))
;  6831     6941  2         ELSE
;  6832     6942  2             ADDRESS_2 = .ADDRESS;
;  6833     6943  2         END;
;  6834     6944  2
;  6835     6945  2
;  6836     6946  2     ! Print the address, but symbolize it first.
;  6837     6947  2     !
;  6838     6948  2     IF .EVENT [EVENT$V_BREAK_ADDRESS]
;  6839     6949  2     THEN
;  6840     6950  2         DBG$PRINT_IDENTIFIER (.EVENT [EVENT$L_PRIMARY])
;  6841     6951  2     ELSE
;  6842     6952  2         DBG$PRINT_IDENTIFIER_PC (.ADDRESS_2);
;  6843     6953  2
;  6844     6954  2
;  6845     6955  2     ! Display the instruction we're at, if we're not line stepping.
;  6846     6956  2     !
;  6847     6957  2     IF .DBG$GB_STP_PTR [EVENT$V_STEPPING_LINE] THEN RETURN;
;  6848     6958  2
;  6849     6959  2     DBG$PRINT($AC (': '));
;  6850     6960  2     DBG$INS_DECODE (.ADDRESS, TRUE, FALSE);
;  6851     6961  1     END;


                                                        .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                    20  65  6E  69  74  75  6F  72  08  00392 P.ADN:  .ASCII  <8>\routine \                    ;
                    20  65  6E  69  74  75  6F  72  08  0039B P.ADO:  .ASCII  <8>\routine \                    ;
```

```
                    20  3A  02  003A4 P.ADP:   .ASCII  <2>\: \                                                 ;

                                                .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                    003C 00000 PRINT_PC_AND_INSTRUCTION:
                                                .WORD   Save R2,R3,R4,R5                                    ; 6912
                    55 00000000G    00   9E 00002          MOVAB   DBG$PRINT, R5                           ;
                    54 00000000'    EF   9E 00009          MOVAB   P.ADN, R4                               ;
                    6D      0000V   CF   9E 00010          MOVAB   PRINT_PC_INST_HANDLER, (FP)             ; 6927
                    52        08    AC   D0 00015          MOVL    EVENT, R2                               ; 6932
                04        19    A2        05 E1 00019      BBC     #5, 25(R2), 1$                          ;
                                          54 DD 0001E      PUSHL   R4                                      ; 6934
                                          14 11 00020      BRB     2$                                      ;
                53        04    AC        02 C3 00022 1$:  SUBL3   #2, ADDRESS, ADDRESS_2                  ; 6938
                                          53 DD 00027      PUSHL   ADDRESS_2                               ;
                    00000000G     00        01 FB 00029      CALLS   #1, DBG$IS_IT_ENTRY                     ;
                                    08      50 E9 00030      BLBC    R0, 3$                                  ;
                                      09    A4 9F 00033      PUSHAB  P.ADO                                   ; 6940
                                    65      01 FB 00036 2$:  CALLS   #1, DBG$PRINT                           ;
                                          04 11 00039      BRB     4$                                      ;
                                    53      04 AC D0 0003B 3$:  MOVL    ADDRESS, ADDRESS_2                  ; 6942
                0C        19    A2        05 E1 0003F 4$:  BBC     #5, 25(R2), 5$                          ; 6948
                                      28    A2 DD 00044      PUSHL   40(R2)                                  ; 6950
                    00000000G     00        01 FB 00047      CALLS   #1, DBG$PRINT_IDENTIFIER                ;
                                          09 11 0004E      BRB     6$                                      ;
                                          53 DD 00050 5$:  PUSHL   ADDRESS_2                               ; 6952
                    00000000G     00        01 FB 00052      CALLS   #1, DBG$PRINT_IDENTIFIER_PC             ;
                                    50 00000000G    00   D0 00059 6$:  MOVL    DBG$GB_STP_PTR, R0            ; 6957
                                    13      01    A0   E8 00060      BLBS    1(R0), 7$                       ;
                                      12    A4 9F 00064      PUSHAB  P.ADP                                   ; 6959
                                    65      01 FB 00067      CALLS   #1, DBG$PRINT                           ;
                                    7E      01 7D 0006A      MOVQ    #1, -(SP)                               ; 6960
                                      04    AC DD 0006D      PUSHL   ADDRESS                                 ;
                            00000000G     00   03 FB 00070      CALLS   #3, DBG$INS_DECODE                  ;
                                          04 00077 7$:     RET                                             ; 6961

; Routine Size:  120 bytes,    Routine Base:  DBG$CODE + 292C
```

```
;  6853          6962  1 ROUTINE PRINT_PC_INST_HANDLER(SIG_ARG_LIST) =
;  6854          6963  1 !++
;  6855          6964  1 ! FUNCTION
;  6856          6965  1 !       Handler for errors that occur during the PRINT_PC_AND_INSTRUCTION
;  6857          6966  1 !       routine. We need a handler so that after displaying the error,
;  6858          6967  1 !       we will unwind to the caller of PRINT_PC_AND_INSTRUCTION, and
;  6859          6968  1 !       continue with any eventpoint processing that was being done.
;  6860          6969  1 !
;  6861          6970  1 ! INPUTS
;  6862          6971  1 !       SIG_ARG_LIST    - The signal argument list that we get from
;  6863          6972  1 !                         the condition handling facility.
;  6864          6973  1 !
;  6865          6974  1 ! OUTPUTS
;  6866          6975  1 !
;  6867          6976  2     BEGIN
;  6868          6977  2     MAP
;  6869          6978  2         SIG_ARG_LIST: REF BLOCK[,BYTE];
;  6870          6979  2
;  6871          6980  2
;  6872          6981  2     ! If informational just resignal
;  6873          6982  2     !
;  6874          6983  2     IF .SIG_ARG_LIST[CHF$L_SIG_NAME]
;  6875          6984  2     THEN
;  6876          6985  2         RETURN SS$_RESIGNAL;
;  6877          6986  2
;  6878          6987  2
;  6879          6988  2     ! If we are in the middle of an unwind then continue unwinding.
;  6880          6989  2     !
;  6881          6990  2     IF .SIG_ARG_LIST[CHF$L_SIG_NAME] EQL SS$_UNWIND
;  6882          6991  2     THEN
;  6883          6992  2         RETURN SS$_CONTINUE;
;  6884          6993  2
;  6885          6994  2
;  6886          6995  2     ! Otherwise, display a message saying that an error occurred during
;  6887          6996  2     ! instruction decode and then display the actual error that occured.
;  6888          6997  2     !
;  6889          6998  2     SIGNAL(DBG$_ERRINSDEC);
;  6890          6999  2     DBG$PUTMSG(.SIG_ARG_LIST);
;  6891          7000  2
;  6892          7001  2
;  6893          7002  2     ! Then unwind to the routine that called PRINT_PC_AND_INSTRUCTION.
;  6894          7003  2     !
;  6895          7004  2     $SETUNWIND();
;  6896          7005  2     RETURN SS$_CONTINUE;
;  6897          7006  1     END;


                                0004 00000 PRINT_PC_INST_HANDLER:
                                           .WORD   Save R2                          ;  6962
                        52  04  AC  D0 00002       MOVL    SIG_ARG_LIST, R2        ;  6983
                        06  04  A2  E9 00006       BLBC    4(R2), 1$
                        50  0918  8F  3C 0C00A     MOVZWL  #2328, R0               ;  6985
                                  04 0000F         RET
                00000920  8F  04  A2  D1 00010 1$: CMPL    4(R2), #2336            ;  6990
```

```
                                 1F  13 00018          BEQL    2$                                      ;
                        000280AB 8F  DD 0001A          PUSHL   #164011                                 ; 6998
                00000000G   00   01  FB 00020          CALLS   #1, LIB$SIGNAL                          ;
                                 52  DD 00027          PUSHL   R2                                      ; 6999
                00000000G   00   01  FB 00029          CALLS   #1, DBG$PUTMSG                          ;
                                 7E  7C 00030          CLRQ    -(SP)                                   ; 7004
                00000000G   00   02  FB 00032          CALLS   #2, SYS$UNWIND                          ;
                            50   01  D0 00039 2$:       MOVL    #1, R0                                  ; 7005
                                 04  00003C           RET                                             ; 7006

; Routine Size:  61 bytes,    Routine Base:  DBG$CODE + 29A4
```

```
;  6899          7007  1 ROUTINE SHOW_EVENT_ENTRY(EVENT_ENTRY): NOVALUE =
;  6900          7008  1 !
;  6901          7009  1 ! FUNCTION
;  6902          7010  1 !       This routine displays an event entry as in:
;  6903          7011  1 !
;  6904          7012  1 !       SHOW [ BREAK | TRACE | WATCH ]
;  6905          7013  1 !
;  6906          7014  1 ! INPUTS
;  6907          7015  1 !       EVENT_ENTRY :           Pointer to event entry.
;  6908          7016  1 !
;  6909          7017  1 ! OUTPUTS
;  6910          7018  1 !       Displays the event entry via DBG$PRINT.
;  6911          7019  1 !
;  6912          7020  1
;  6913          7021  2     BEGIN
;  6914          7022  2
;  6915          7023  2     MAP
;  6916          7024  2         EVENT_ENTRY: REF EVENT$EVENT_DESCRIPTOR; ! Event Entry Pointer
;  6917          7025  2
;  6918          7026  2
;  6919          7027  2     LOCAL
;  6920          7028  2         VMS_DESC: DBG$STG_DESC,             !
;  6921          7029  2         VAL_DESC: REF DBG$VALDESC,          !
;  6922          7030  2         WHEN_ENTRY:                         ! When Entry Pointer
;  6923          7031  2             REF EVENT$WHEN_DESCRIPTOR,!
;  6924          7032  2         DO_LIST_ENTRY:                      ! DO List Entry Pointer
;  6925          7033  2             REF EVENT$DO_LIST_DESCRIPTOR,!
;  6926          7034  2         STRING_DESCRIPTOR: BLOCK[8,BYTE],! String descriptor
;  6927          7035  2         OUTPUT_BUFFER:                      ! Temporary formatting buffer
;  6928          7036  2             VECTOR[TTY_OUT_WIDTH,BYTE],!
;  6929          7037  2         DUMMY;                              ! Dummy routine argument
;  6930          7038  2
;  6931          7039  2
;  6932          7040  2
;  6933          7041  2     ! Skip STEPS, SKIPS, and IGNOR entries.
;  6934          7042  2     !
;  6935          7043  2     IF .EVENT_ENTRY [EVENT$B_CMD_TYPE] EQLU EVENT$K_TYPE_STEPS OR
;  6936          7044  2        .EVENT_ENTRY [EVENT$B_CMD_TYPE] EQLU EVENT$K_TYPE_SKIPS OR
;  6937          7045  2        .EVENT_ENTRY [EVENT$B_CMD_TYPE] EQLU EVENT$K_TYPE_IGNOR OR
;  6938          7046  2        .EVENT_ENTRY [EVENT$B_CMD_TYPE] EQLU EVENT$K_TYPE_SSINT
;  6939          7047  2     THEN
;  6940          7048  2         RETURN;
;  6941          7049  2
;  6942          7050  2
;  6943          7051  2     ! Setup a string descriptor.
;  6944          7052  2     !
;  6945          7053  2     STRING_DESCRIPTOR [DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  6946          7054  2     STRING_DESCRIPTOR [DSC$B_CLASS] = DSC$K_CLASS_S;
;  6947          7055  2     STRING_DESCRIPTOR [DSC$A_POINTER] = 0;
;  6948          7056  2     STRING_DESCRIPTOR [DSC$W_LENGTH] = 0;
;  6949          7057  2
;  6950          7058  2
;  6951          7059  2     ! For watchpoint events, see whether the watched Primary
;  6952          7060  2     ! is still valid. If not, convert the Primary to a Volatile
;  6953          7061  2     ! Value Descriptor and display an informational to that effect.
;  6954          7062  2     !
;  6955          7063  2     IF .EVENT_ENTRY[EVENT$B_SUB_KIND] EQL EVENT$K_ACC_MDFY
```

```
;  6956     7064  2     THEN
;  6957     7065  2         BEGIN
;  6958     7066  2         LOCAL
;  6959     7067  2             PRIMARY,
;  6960     7068  2             SYMID_LIST,
;  6961     7069  2             VALDESC,
;  6962     7070  2             V_VALDESC: REF DBG$VALDESC;
;  6963     7071  2
;  6964     7072  2
;  6965     7073  2         ! Copy the Primary since PRIM_TO_VAL sometimes has the side effect
;  6966     7074  2         ! of modifying its argument. This copy is done into temporary
;  6967     7075  2         ! memory (fourth parameter is FALSE). Then call PRIM_TO_VAL
;  6968     7076  2         ! and get a Volatile Value Descriptor (this will have
;  6969     7077  2         ! the real address of the watched variable embedded in it).
;  6970     7078  2         ! PRIM_TO_VAL is a cover routine for DBG$PRIM_TO_VAL; it has
;  6971     7079  2         ! a handler which sets INVALID_FLAG if the Primary is no longer
;  6972     7080  2         ! valid (has gone out of scope).
;  6973     7081  2         !
;  6974     7082  2         DBG$NCOPY_DESC (.EVENT_ENTRY[EVENT$L_PRIMARY], PRIMARY, DUMMY, FALSE);
;  6975     7083  2         INVALID_FLAG = FALSE;
;  6976     7084  2         PRIM_TO_VAL (.PRIMARY, DBG$K_V_VALUE_DESC, V_VALDESC);
;  6977     7085  2
;  6978     7086  2
;  6979     7087  2         ! Check whether the Primary is still valid, i.e., whether it still
;  6980     7088  2         ! points to the same storage as it did when the watchpoint was set.
;  6981     7089  2         !
;  6982     7090  2         IF .INVALID_FLAG OR
;  6983     7091  2            CH$NEQ (T2, V_VALDESC[DBG$A_VALUE_VMSDESC],
;  6984     7092  2                    12, EVENT_ENTRY[EVENT$A_VMSDESC])
;  6985     7093  2         THEN
;  6986     7094  4             BEGIN
;  6987     7095  4
;  6988     7096  4
;  6989     7097  4             ! Print an informational saying the Primary is invalid.
;  6990     7098  4             !
;  6991     7099  4             DBG$PRINT ($AC ('%DEBUG-I-WATCHVAR, watched variable '));
;  6992     7100  4             DBG$PRINT_IDENTIFIER(.EVENT_ENTRY[EVENT$L_PRIMARY]);
;  6993     7101  4             IF .INVALID_FLAG
;  6994     7102  4             THEN
;  6995     7103  4                 DBG$PRINT ($AC (' has gone out of scope'))
;  6996     7104  4             ELSE
;  6997     7105  4                 DBG$PRINT ($AC (' now points to a different address'));
;  6998     7106  4             DBG$NEWLINE();
;  6999     7107  4
;  7000     7108  4
;  7001     7109  4             ! Free up the Primary.
;  7002     7110  4             !
;  7003     7111  4             IF DBG$NGET_SYMID (.EVENT_ENTRY[EVENT$L_PRIMARY], SYMID_LIST, DUMMY)
;  7004     7112  4             THEN
;  7005     7113  4                 DBG$STA_UNLOCK_SYMID (.SYMID_LIST);
;  7006     7114  4             DBG$NFREE_DESC (.EVENT_ENTRY[EVENT$L_PRIMARY], DUMMY);
;  7007     7115  4
;  7008     7116  4
;  7009     7117  4             ! Replace it with the Volatile Value Descriptor.
;  7010     7118  4             !
;  7011     7119  4             V_VALDESC = DBG$MAKE_VAL_DESC(EVENT_ENTRY[EVENT$A_VMSDESC], DBG$K_V_VALUE_DESC);
;  7012     7120  4             IF DBG$NGET_SYMID (.V_VALDESC, SYMID_LIST, DUMMY)
```

```
;  7013          7121  4            THEN
;  7014          7122  4                DBG$STA_LOCK_SYMID (.SYMID_LIST);
;  7015          7123  4            DBG$NCOPY_DESC (.V_VALDESC, EVENT_ENTRY[EVENT$L_PRIMARY], DUMMY, TRUE);
;  7016          7124  3            END;
;  7017          7125  2        END;
;  7018          7126  2
;  7019          7127  2
;  7020          7128  2        ! Print the event type: breakpoint, tracepoint, or watchpoint.
;  7021          7129  2        !
;  7022          7130  2        SELECTONE .EVENT_ENTRY[EVENT$B_CMD_TYPE] OF
;  7023          7131  2            SET
;  7024          7132  2
;  7025          7133  2            [EVENT$K_TYPE_BREAK]:
;  7026          7134  2                DBG$PRINT($AC('breakpoint'));
;  7027          7135  2
;  7028          7136  2            [EVENT$K_TYPE_TRACE]:
;  7029          7137  2                DBG$PRINT($AC('tracepoint'));
;  7030          7138  2
;  7031          7139  2            [EVENT$K_TYPE_WATCH]:
;  7032          7140  2                DBG$PRINT($AC('watchpoint'));
;  7033          7141  2
;  7034          7142  2            TES;
;  7035          7143  2
;  7036          7144  2
;  7037          7145  2        ! Display the event kind (access, exception, instruction).
;  7038          7146  2        !
;  7039          7147  2        SELECTONE .EVENT_ENTRY [EVENT$B_CMD_KIND] OF
;  7040          7148  2            SET
;  7041          7149  2
;  7042          7150  2
;  7043          7151  2        ! This is an access (read, write, modify, execute) kind
;  7044          7152  2        ! of event point.  Display the address.
;  7045          7153  2        !
;  7046          7154  2        [EVENT$K_KIND_ACC]:
;  7047          7155  3            BEGIN
;  7048          7156  3
;  7049          7157  3
;  7050          7158  3            ! Print the access kind if read, write, or modify.  If the address
;  7051          7159  3            ! is an entry point, indicate that it is a routine.
;  7052          7160  3            !
;  7053          7161  3            IF .EVENT_ENTRY[EVENT$B_CMD_TYPE] NEQU EVENT$K_TYPE_WATCH
;  7054          7162  3            THEN
;  7055          7163  4                BEGIN
;  7056          7164  4                SELECTONE .EVENT_ENTRY[EVENT$B_SUB_KIND] OF
;  7057          7165  4                    SET
;  7058          7166  4
;  7059          7167  4                    [EVENT$K_ACC_READ]:
;  7060          7168  4                        DBG$PRINT($AC(' on read of '));
;  7061          7169  4
;  7062          7170  4                    [EVENT$K_ACC_WRIT]:
;  7063          7171  4                        DBG$PRINT($AC(' on write of '));
;  7064          7172  4
;  7065          7173  4                    [EVENT$K_ACC_MDFY]:
;  7066          7174  4                        DBG$PRINT($AC(' on modify of '));
;  7067          7175  4
;  7068          7176  4                    [EVENT$K_ACC_EXEC]:
;  7069          7177  4                        DBG$PRINT($AC(' at '));
```

```
;  7070     7178  4                    [EVENT$K_ACC_RTRN]:
;  7071     7179  4                        DBG$PRINT($AC(' on return from '));
;  7072     7180  4
;  7073     7181  4                    TES;
;  7074     7182  4
;  7075     7183  4
;  7076     7184  4                END
;  7077     7185  4
;  7078     7186  4
;  7079     7187  4            ! For watchpoints, simply say "watchpoint of".
;  7080     7188  4            !
;  7081     7189  3            ELSE
;  7082     7190  3                DBG$PRINT($AC (' of '));
;  7083     7191  3
;  7084     7192  3
;  7085     7193  3            ! We print the address next.  If the address is a routine address
;  7086     7194  3            ! (a CALLS/CALLG entry point), we refer to it as "routine XXX",
;  7087     7195  3            ! where XXX is the routine name.  If it is not a routine, we omit
;  7088     7196  3            ! the "routine" designation.
;  7089     7197  3            !
;  7090     7198  3            IF .EVENT_ENTRY[EVENT$V_BREAK_ADDRESS] OR
;  7091     7199  3               DBG$IS_IT_ENTRY(.EVENT_ENTRY[EVENT$L_ADDRESS] - 2)
;  7092     7200  3            THEN
;  7093     7201  3                DBG$PRINT($AC('routine '));
;  7094     7202  3
;  7095     7203  3            DBG$PRINT_IDENTIFIER(.EVENT_ENTRY[EVENT$L_PRIMARY]);
;  7096     7204  3
;  7097     7205  3
;  7098     7206  3            ! Check for the /SOURCE or /NOSOURCE override switch.  If set, we
;  7099     7207  3            ! print "[source]" or "[nosource]", as appropriate.
;  7100     7208  3            !
;  7101     7209  3            IF .EVENT_ENTRY[EVENT$V_OVRD_SOURCE]
;  7102     7210  3            THEN
;  7103     7211  4                BEGIN
;  7104     7212  4                IF .EVENT_ENTRY[EVENT$V_STEP_SOURCE]
;  7105     7213  4                THEN
;  7106     7214  4                    DBG$PRINT($AC(' [source]'))
;  7107     7215  4
;  7108     7216  4                ELSE
;  7109     7217  4                    DBG$PRINT($AC(' [nosource]'));
;  7110     7218  4
;  7111     7219  3                END;
;  7112     7220  3
;  7113     7221  2            END;                        ! End of case for access kind of event
;  7114     7222  2
;  7115     7223  2
;  7116     7224  2        ! This is an instruction type.
;  7117     7225  2        !
;  7118     7226  2        [EVENT$K_KIND_INS]:
;  7119     7227  3            BEGIN
;  7120     7228  3
;  7121     7229  3
;  7122     7230  3            ! Display the /SOURCE or /NOSOURCE qualifier if specified.
;  7123     7231  3            !
;  7124     7232  3            IF .EVENT_ENTRY[EVENT$V_OVRD_SOURCE]
;  7125     7233  3            THEN
;  7126     7234  4                BEGIN
```

```
;  7127     7235  4              IF .EVENT_ENTRY[EVENT$V_STEP_SOURCE]
;  7128     7236  4              THEN
;  7129     7237  4                  DBG$PRINT($AC('/source'))
;  7130     7238  4
;  7131     7239  4              ELSE
;  7132     7240  4                  DBG$PRINT($AC('/nosource'));
;  7133     7241  4
;  7134     7242  3              END;
;  7135     7243  3
;  7136     7244  3
;  7137     7245  3          ! Display the /SOURCE or /NOSOURCE qualifier if specified.
;  7138     7246  3          !
;  7139     7247  3          IF .EVENT_ENTRY[EVENT$V_OVRD_NOSYSTEM]
;  7140     7248  3          THEN
;  7141     7249  4              BEGIN
;  7142     7250  4              IF .EVENT_ENTRY[EVENT$V_STEP_NOSYSTEM]
;  7143     7251  4              THEN
;  7144     7252  4                  DBG$PRINT($AC('/nosystem'))
;  7145     7253  4
;  7146     7254  4              ELSE
;  7147     7255  4                  DBG$PRINT($AC('/system'));
;  7148     7256  4
;  7149     7257  3              END;
;  7150     7258  3
;  7151     7259  3
;  7152     7260  3          ! Display the /OVER or /INTO qualifier if specified.
;  7153     7261  3          !
;  7154     7262  3          IF .EVENT_ENTRY[EVENT$V_OVRD_OVER]
;  7155     7263  3          THEN
;  7156     7264  4              BEGIN
;  7157     7265  4              IF .EVENT_ENTRY[EVENT$V_STEP_OVER]
;  7158     7266  4              THEN
;  7159     7267  4                  DBG$PRINT($AC('/over'))
;  7160     7268  4
;  7161     7269  4              ELSE
;  7162     7270  4                  DBG$PRINT($AC('/into'));
;  7163     7271  4
;  7164     7272  3              END;
;  7165     7273  3
;  7166     7274  3
;  7167     7275  3          ! Display the instruction type (calls, branches, all, or user
;  7168     7276  3          ! list.
;  7169     7277  3          !
;  7170     7278  3          DBG$PRINT   ($AC     (' on !AC'),
;  7171     7279  3                       SELECTONE .EVENT_ENTRY [EVENT$B_SUB_KIND] OF
;  7172     7280  3                           SET
;  7173     7281  3                           [EVENT$K_INS_CALL]:
;  7174     7282  3                               $AC ('calls:');
;  7175     7283  3                           [EVENT$K_INS_BRAN]:
;  7176     7284  3                               $AC ('branches:');
;  7177     7285  3                           [EVENT$K_INS_EVRY]:
;  7178     7286  3                               $AC ('instructions');
;  7179     7287  3                           [EVENT$K_INS_USER]:
;  7180     7288  3                               $AC ('instruction(s):');
;  7181     7289  3                           [EVENT$K_INS_LINE]:
;  7182     7290  3                               $AC ('lines');
;  7183     7291  3                           TES
```

```
;  7184          7292  3                        );
;  7185          7293  3
;  7186          7294  2                    END;
;  7187          7295  2
;  7188          7296  2
;  7189          7297  2                ! This is an exception type.
;  7190          7298  2                !
;  7191          7299  2                [EVENT$K_KIND_EXC]:
;  7192          7300  2                    DBG$PRINT($AC(' on exception'));
;  7193          7301  2
;  7194          7302  2                TES;
;  7195          7303  2
;  7196          7304  2
;  7197          7305  2            ! If this entry is silent, say so.
;  7198          7306  2            !
;  7199          7307  2            IF .EVENT_ENTRY[EVENT$V_SILENT] THEN DBG$PRINT($AC(' [silent]'));
;  7200          7308  2
;  7201          7309  2
;  7202          7310  2            ! If this entry is temporary, say so.
;  7203          7311  2            !
;  7204          7312  2            IF .EVENT_ENTRY[EVENT$V_ONCE_ONLY] THEN DBG$PRINT($AC(' [temporary]'));
;  7205          7313  2
;  7206          7314  2
;  7207          7315  2            ! If this entry is an instruction type (/CALL, /BRANCH, etc.) and
;  7208          7316  2            ! has an opcode bit map (i.e., is not for all instructions),
;  7209          7317  2            ! display the opcode mnemonics indicated.
;  7210          7318  2            !
;  7211          7319  2            IF .EVENT_ENTRY [EVENT$B_CMD_KIND] EQLU EVENT$K_KIND_INS AND
;  7212          7320  2               .EVENT_ENTRY [EVENT$L_OPCODE_LIST] NEQA 0
;  7213          7321  2            THEN
;  7214          7322  2                BEGIN
;  7215          7323  3
;  7216          7324  3
;  7217          7325  3                ! Declare a pointer to an opcode list, and an opcode
;  7218          7326  3                ! variable.
;  7219          7327  3                !
;  7220          7328  3                LOCAL
;  7221          7329  3                    OPCODE_LIST: REF BITVECTOR [512],
;  7222          7330  3                    COLUMN;
;  7223          7331  3
;  7224          7332  3
;  7225          7333  3                ! Point to the opcode list, and set the column to 0.
;  7226          7334  3                !
;  7227          7335  3                OPCODE_LIST = .EVENT_ENTRY [EVENT$L_OPCODE_LIST];
;  7228          7336  3                COLUMN = -1;
;  7229          7337  3                INCR OPCODE FROM 0 TO 511 DO
;  7230          7338  4                    BEGIN
;  7231          7339  4                    IF .OPCODE_LIST [.OPCODE] THEN
;  7232          7340  5                        BEGIN
;  7233          7341  5                        LOCAL INDEX;
;  7234          7342  5                        COLUMN = (.COLUMN + 1) AND 7;
;  7235          7343  5                        IF .COLUMN EQLU 0 THEN DBG$NEWLINE();
;  7236          7344  5                        INDEX = .DBG$OPCODE_KIND_TABLE[.OPCODE];
;  7237          7345  5                        DBG$PRINT($AC ('!: !AD'), 6,
;  7238          7346  5                                              DBG$OPCODE_NAME_TABLE[.INDEX,0,0,0,0]);
;  7239          7347  4                        END;
;  7240          7348  4
```

```
;  7241     7349  3              END;
;  7242     7350  3
;  7243     7351  3          DBG$NEWLINE();
;  7244     7352  3          END
;  7245     7353  3
;  7246     7354  3
;  7247     7355  3      ! Otherwise, just print the line.
;  7248     7356  3      !
;  7249     7357  3      ELSE
;  7250     7358  3          DBG$NEWLINE ();
;  7251     7359  3
;  7252     7360  3
;  7253     7361  3      ! If the after count is not the default (1), print it.
;  7254     7362  3      !
;  7255     7363  3      IF .EVENT_ENTRY [EVENT$L_AFTER_COUNT] NEQU 1
;  7256     7364  3      THEN
;  7257     7365  3          BEGIN
;  7258     7366  3          DBG$PRINT($AC('    /after: !UL'), .EVENT_ENTRY[EVENT$L_AFTER_COUNT]);
;  7259     7367  3          DBG$NEWLINE();
;  7260     7368  3          END;
;  7261     7369  3
;  7262     7370  3
;  7263     7371  3      ! If there is a WHEN entry, print it.
;  7264     7372  3      !
;  7265     7373  3      IF .EVENT_ENTRY [EVENT$L_WHEN] NEQA 0
;  7266     7374  3      THEN
;  7267     7375  3          BEGIN
;  7268     7376  3          LOCAL
;  7269     7377  3              STRING :    REF VECTOR [, WORD];
;  7270     7378  3
;  7271     7379  3          WHEN_ENTRY = .EVENT_ENTRY [EVENT$L_WHEN];
;  7272     7380  3          STRING = .WHEN_ENTRY [EVENT$L_WHEN_POINT];
;  7273     7381  3          DBG$PRINT($AC('    when (!AD)'), .STRING[0] - 1, STRING[1]);
;  7274     7382  3          DBG$NEWLINE();
;  7275     7383  3          END;
;  7276     7384  3
;  7277     7385  3
;  7278     7386  3      ! If there is a DO list entry, print it.
;  7279     7387  3      !
;  7280     7388  3      IF .EVENT_ENTRY [EVENT$L_DO] NEQA 0
;  7281     7389  3      THEN
;  7282     7390  3          BEGIN
;  7283     7391  3          LOCAL
;  7284     7392  3              STRING :    REF VECTOR [, WORD];
;  7285     7393  3
;  7286     7394  3          DO_LIST_ENTRY = .EVENT_ENTRY [EVENT$L_DO];
;  7287     7395  3          STRING = .DO_LIST_ENTRY [EVENT$L_DO_LIST_POINT];
;  7288     7396  3          DBG$PRINT($AC('    do (!AD)'), .STRING[0] - 1, STRING[1]);
;  7289     7397  3          DBG$NEWLINE();
;  7290     7398  3          END;
;  7291     7399  3
;  7292     7400  1      END;


                                                .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0
```

```
48 43 54 41 57 2D 49 2D 47 55 42 45 44 25 24 003A7 P.ADQ:  .ASCII  \$%DEBUG-I-WATCHVAR, watched variable \
61 76 20 64 65 68 63 74 61 77 20 2C 52 41 56 003B6
                        20 65 6C 62 61 69 72 003C5
20 74 75 6F 20 65 6E 6F 67 20 73 61 68 20 16 003CC P.ADR:  .ASCII  <22>\ has gone out of scope\
                        65 70 6F 63 73 20 66 6F 003DB
6F 74 20 73 74 6E 69 6F 70 20 77 6F 6E 20 22 003E3 P.ADS:  .ASCII  \" now points to a different address\
64 61 20 74 6E 65 72 65 66 66 69 64 20 61 20 003F2
                              73 73 65 72 64 00401
         74 6E 69 6F 70 6B 61 65 72 62 0A 00406 P.ADT:  .ASCII  <10>\breakpoint\
         74 6E 69 6F 70 65 63 61 72 74 0A 00411 P.ADU:  .ASCII  <10>\tracepoint\
         74 6E 69 6F 70 68 63 74 61 77 0A 0041C P.ADV:  .ASCII  <10>\watchpoint\
      20 66 6F 20 64 61 65 72 20 6E 6F 20 0C 00427 P.ADW:  .ASCII  <12>\ on read of \
   20 66 6F 20 65 74 69 72 77 20 6E 6F 20 0D 00434 P.ADX:  .ASCII  <13>\ on write of \
20 66 6F 20 79 66 69 64 6F 6D 20 6E 6F 20 0E 00442 P.ADY:  .ASCII  <14>\ on modify of \
                              20 74 61 20 04 00451 P.ADZ:  .ASCII  <4>\ at \
6F 72 66 20 6E 72 75 74 65 72 20 6E 6F 20 10 00456 P.AEA:  .ASCII  <16>\ on return from \
                                       20 6D 00465
                              20 66 6F 20 04 00467 P.AEB:  .ASCII  <4>\ of \
            20 65 6E 69 74 75 6F 72 08 0046C P.AEC:  .ASCII  <8>\routine \
         5D 65 63 72 75 6F 73 5B 20 09 00475 P.AED:  .ASCII  <9>\ [source]\
   5D 65 63 72 75 6F 73 6F 6E 5B 20 0B 0047F P.AEE:  .ASCII  <11>\ [nosource]\
            65 63 72 75 6F 73 2F 07 0048B P.AEF:  .ASCII  <7>\/source\
      65 63 72 75 6F 73 6F 6E 2F 09 00493 P.AEG:  .ASCII  <9>\/nosource\
      6D 65 74 73 79 73 6F 6E 2F 09 0049D P.AEH:  .ASCII  <9>\/nosystem\
            6D 65 74 73 79 73 2F 07 004A7 P.AEI:  .ASCII  <7>\/system\
                  72 65 76 6F 2F 05 004AF P.AEJ:  .ASCII  <5>\/over\
                  6F 74 6E 69 2F 05 004B5 P.AEK:  .ASCII  <5>\/into\
            43 41 21 20 6E 6F 20 07 004BB P.AEL:  .ASCII  <7>\ on !AC\
               3A 73 6C 6C 61 63 06 004C3 P.AEM:  .ASCII  <6>\calls:\
      3A 73 65 68 63 6E 61 72 62 09 004CA P.AEN:  .ASCII  <9>\branches:\
   73 6E 6F 69 74 63 75 72 74 73 6E 69 0C 004D4 P.AEO:  .ASCII  <12>\instructions\
29 73 28 6E 6F 69 74 63 75 72 74 73 6E 69 0F 004E1 P.AEP:  .ASCII  <15>\instruction(s):\
                                          3A 004F0
                  73 65 6E 69 6C 05 004F1 P.AEQ:  .ASCII  <5>\lines\
   6E 6F 69 74 70 65 63 78 65 20 6E 6F 20 0D 004F7 P.AER:  .ASCII  <13>\ on exception\
         5D 74 6E 65 6C 69 73 5B 20 09 00505 P.AES:  .ASCII  <9>\ [silent]\
   5D 79 72 61 72 6F 70 6D 65 74 5B 20 0C 0050F P.AET:  .ASCII  <12>\ [temporary]\
                  44 41 21 5F 21 05 0051C P.AEU:  .ASCII  <5>\!_!AD\
4C 55 21 20 3A 72 65 74 66 61 2F 20 20 20 0E 00522 P.AEV:  .ASCII  <14>\   /after: !UL\
   29 44 41 21 28 20 6E 65 68 77 20 20 20 0D 00531 P.AEW:  .ASCII  <13>\   when (!AD)\
      29 44 41 21 28 20 6F 64 20 20 20 0B 0053F P.AEX:  .ASCII  <11>\   do (!AD)\


                                                           .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                           0FFC 00000 SHOW_EVENT_ENTRY:
                                                           .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11     ; 7007
                    5B 00000000G  00  9E 00002      MOVAB   DBG$NGET_SYMID, R11                   ;
                    5A 00000000G  00  9E 00009      MOVAB   DBG$PRINT_IDENTIFIER, R10              ;
                    59 00000000'  EF  9E 00010      MOVAB   INVALID_FLAG, R9                       ;
                    58 00000000G  00  9E 00017      MOVAB   DBG$NEWLINE, R8                        ;
                    57 00000000G  00  9E 0001E      MOVAB   DBG$PRINT, R7                          ;
                    56 00000000'  EF  9E 00025      MOVAB   P.ADQ, R6                              ;
                    5E       FF58  CE  9E 0002C      MOVAB   -168(SP), SP                           ;
                    54         04  AC  D0 00031      MOVL    EVENT_ENTRY, R4                        ; 7043
                    55         14  A4  9E 00035      MOVAB   20(R4), R5                             ;
```

```
                                03              65  91 00039          CMPB    (R5), #3                                ;
                                                0D  13 0003C          BEQL    1$                                      ;
                                04              65  91 0003E          CMPB    (R5), #4                                ; 7044
                                                08  13 00041          BEQL    1$                                      ;
                                05              65  91 00043          CMPB    (R5), #5                                ; 7045
                                                03  13 00046          BEQL    1$                                      ;
                                06              65  91 00048          CMPB    (R5), #6                                ; 7046
                                                01  12 0004B 1$:       BNEQ    2$                                      ;
                                                    04 0004D          RET                                             ;
                            EC  AD 010E0000     8F  D0 0004E 2$:      MOVL    #17694720, STRING_DESCRIPTOR            ; 7056
                                          F0    AD  D4 00056          CLRL    STRING_DESCRIPTOR+4                     ; 7055
                                02        02    A5  91 00059          CMPB    2(R5), #2                               ; 7063
                                                03  13 0005D          BEQL    3$                                      ;
                                                00B1 31 0005F         BRW     9$                                      ;
                                                7E  D4 00062 3$:      CLRL    -(SP)                                   ; 7082
                                          10    AE  9F 00064          PUSHAB  DUMMY                                   ;
                                          08    AE  9F 00067          PUSHAB  PRIMARY                                 ;
                                          28    A4  DD 0006A          PUSHL   40(R4)                                  ;
                        00000000G         00        04 FB 0006D       CALLS   #4, DBG$NCOPY_DESC                      ;
                                                  69 D4 00074         CLRL    INVALID_FLAG                            ; 7083
                                          04    AE  9F 00076          PUSHAB  V_VALDESC                               ; 7084
                                7E        83  8F  9A 00079          MOVZBL  #131, -(SP)                             ;
                                          08    AE  DD 0007D          PUSHL   PRIMARY                                 ;
                            0000V       CF        03 FB 00080       CALLS   #3, PRIM_TO_VAL                         ;
                                        0C        69 E8 00085       BLBS    INVALID_FLAG, 4$                        ; 7090
                                        50  04    AE D0 00088       MOVL    V_VALDESC, R0                           ; 7091
                34  A4          14      A0        0C 29 0008C       CMPC3   #12, 20(R0), 52(R4)                     ; 7092
                                                  7F 13 00092       BEQL    9$                                      ;
                                                  56 DD 00094 4$:   PUSHL   R6                                      ; 7099
                                67                01 FB 00096       CALLS   #1, DBG$PRINT                           ;
                                          28    A4  DD 00099          PUSHL   40(R4)                                  ; 7100
                                6A                01 FB 0009C       CALLS   #1, DBG$PRINT_IDENTIFIER                ;
                                05                69 E9 0009F       BLBC    INVALID_FLAG, 5$                        ; 7101
                                          25    A6  9F 000A2          PUSHAB  P.ADR                                   ; 7103
                                                03  11 000A5         BRB     6$                                      ;
                                          3C    A6  9F 000A7 5$:      PUSHAB  P.ADS                                   ; 7105
                                67                01 FB 000AA 6$:       CALLS   #1, DBG$PRINT                           ;
                                68                00 FB 000AD       CALLS   #0, DBG$NEWLINE                         ; 7106
                                          0C    AE  9F 000B0          PUSHAB  DUMMY                                   ; 7111
                                          0C    AE  9F 000B3          PUSHAB  SYMID_LIST                              ;
                                          28    A4  DD 000B6          PUSHL   40(R4)                                  ;
                                6B                03 FB 000B9       CALLS   #3, DBG$NGET_SYMID                      ;
                                0A                50 E9 000BC       BLBC    R0, 7$                                  ;
                                          08    AE  DD 000BF          PUSHL   SYMID_LIST                              ; 7113
                        00000000G         00        01 FB 000C2       CALLS   #1, DBG$STA_UNLOCK_SYMID                ;
                                          0C    AE  9F 000C9 7$:      PUSHAB  DUMMY                                   ; 7114
                                          28    A4  DD 000CC          PUSHL   40(R4)                                  ;
                        00000000G         00        02 FB 000CF       CALLS   #2, DBG$NFREE_DESC                      ;
                                7E        83  8F  9A 000D6          MOVZBL  #131, -(SP)                             ; 7119
                                          34    A4  9F 000DA          PUSHAB  52(R4)                                  ;
                        00000000G         00        02 FB 000DD       CALLS   #2, DBG$MAKE_VAL_DESC                   ;
                                    04    AE        50 D0 000E4       MOVL    R0, V_VALDESC                           ;
                                          0C    AE  9F 000E8          PUSHAB  DUMMY                                   ; 7120
                                          0C    AE  9F 000EB          PUSHAB  SYMID_LIST                              ;
                                          0C    AE  DD 000EE          PUSHL   V_VALDESC                               ;
                                6B                03 FB 000F1       CALLS   #3, DBG$NGET_SYMID                      ;
                                0A                50 E9 000F4       BLBC    R0, 8$                                  ;
```

```
                                 08   AE  DD 000F7           PUSHL   SYMID_LIST                      ; 7122
          00000000G  00               01  FB 000FA           CALLS   #1, DBG$STA_LOCK_SYMID
                                      01  DD 00101  8$:      PUSHL   #1                              ; 7123
                                 10   AE  9F 00103           PUSHAB  DUMMY
                                 28   A4  9F 00106           PUSHAB  40(R4)
                                 10   AE  DD 00109           PUSHL   V_VALDESC
          00000000G  00               04  FB 0010C           CALLS   #4, DBG$NCOPY_DESC
                                      65  95 00113  9$:      TSTB    (R5)                            ; 7133
                                      05  12 00115           BNEQ    10$
                                 5F   A6  9F 00117           PUSHAB  P.ADT                           ; 7134
                                      12  11 0011A           BRB     12$
                     01               65  91 0011C  10$:     CMPB    (R5), #1                        ; 7136
                                      05  12 0011F           BNEQ    11$
                                 6A   A6  9F 00121           PUSHAB  P.ADU                           ; 7137
                                      08  11 00124           BRB     12$
                     02               65  91 00126  11$:     CMPB    (R5), #2                        ; 7139
                                      06  12 00129           BNEQ    13$
                                 75   A6  9F 0012B           PUSHAB  P.ADV                           ; 7140
                     67               01  FB 0012E  12$:     CALLS   #1, DBG$PRINT
                                 01   A5  95 00131  13$:     TSTB    1(R5)                           ; 7154
                                      03  13 00134           BEQL    14$
                                    0082  31 00136           BRW     27$
                     02               65  91 00139  14$:     CMPB    (R5), #2                        ; 7161
                                      3B  13 0013C           BEQL    19$
                                 02   A5  95 0013E           TSTB    2(R5)                           ; 7167
                                      06  12 00141           BNEQ    15$
                               0080   C6  9F 00143           PUSHAB  P.ADW                           ; 7168
                                      34  11 00147           BRB     20$
                     01          02   A5  91 00149  15$:     CMPB    2(R5), #1                       ; 7170
                                      06  12 0014D           BNEQ    16$
                               008D   C6  9F 0014F           PUSHAB  P.ADX                           ; 7171
                                      28  11 00153           BRB     20$
                     02          02   A5  91 00155  16$:     CMPB    2(R5), #2                       ; 7173
                                      06  12 00159           BNEQ    17$
                               009B   C6  9F 0015B           PUSHAB  P.ADY                           ; 7174
                                      1C  11 0015F           BRB     20$
                     03          02   A5  91 00161  17$:     CMPB    2(R5), #3                       ; 7176
                                      06  12 00165           BNEQ    18$
                               00AA   C6  9F 00167           PUSHAB  P.ADZ                           ; 7177
                                      10  11 0016B           BRB     20$
                     04          02   A5  91 0016D  18$:     CMPB    2(R5), #4                       ; 7179
                                      0D  12 00171           BNEQ    21$
                               00AF   C6  9F 00173           PUSHAB  P.AEA                           ; 7180
                                      04  11 00177           BRB     20$
                               00C0   C6  9F 00179  19$:     PUSHAB  P.AEB                           ; 7190
                     67               01  FB 0017D  20$:     CALLS   #1, DBG$PRINT
0F             19    A4               05  E0 00180  21$:     BBS     #5, 25(R4), 22$                 ; 7198
7E             2C    A4               02  C3 00185           SUBL3   #2, 44(R4), -(SP)               ; 7199
          00000000G  00               01  FB 0018A           CALLS   #1, DBG$IS_IT_ENTRY
                     07               50  E9 00191           BLBC    R0, 23$
                               00C5   C6  9F 00194  22$:     PUSHAB  P.AEC                           ; 7201
                     67               01  FB 00198           CALLS   #1, DBG$PRINT
                                 28   A4  DD 0019B  23$:     PUSHL   40(R4)                          ; 7203
                     6A               01  FB 0019E           CALLS   #1, DBG$PRINT_IDENTIFIER
03             18    A4               04  E0 001A1           BBS     #4, 24(R4), 24$                 ; 7209
                                    00BA  31 001A6           BRW     47$
06             18    A4               05  E1 001A9  24$:     BBC     #5, 24(R4), 25$                 ; 7212
```

```
                00CE   C6  9F 001AE          PUSHAB  P.AED                           ; 7214
                       04  11 001B2          BRB     26$                             ;
                00D8   C6  9F 001B4 25$:     PUSHAB  P.AEE                           ; 7217
                00A5   31  11 001B8 26$:     BRW     46$                             ;
          01      01   A5  91 001BB 27$:     CMPB    1(R5), #1                       ; 7226
                       03  13 001BF          BEQL    28$                             ;
                0092   31  11 001C1          BRW     45$                             ;
          52      18   A4  9E 001C4 28$:     MOVAB   24(R4), R2                      ; 7232
    11    62           04  E1 001C8          BBC     #4, (R2), 31$                   ;
    06    62           05  E1 001CC          BBC     #5, (R2), 29$                   ; 7235
                00E4   C6  9F 001D0          PUSHAB  P.AEF                           ; 7237
                       04  11 001D4          BRB     30$                             ;
                00EC   C6  9F 001D6 29$:     PUSHAB  P.AEG                           ; 7240
          67           01  FB 001DA 30$:     CALLS   #1, DBG$PRINT                   ;
    11    62           06  E1 001DD 31$:     BBC     #6, (R2), 34$                   ; 7247
                       62  95 001E1          TSTB    (R2)                            ; 7250
                       06  18 001E3          BGEQ    32$                             ;
                00F6   C6  9F 001E5          PUSHAB  P.AEH                           ; 7252
                       04  11 001E9          BRB     33$                             ;
                0100   C6  9F 001EB 32$:     PUSHAB  P.AEI                           ; 7255
          67           01  FB 001EF 33$:     CALLS   #1, DBG$PRINT                   ;
    11    62           02  E1 001F2 34$:     BBC     #2, (R2), 37$                   ; 7262
    06    62           03  E1 001F6          BBC     #3, (R2), 35$                   ; 7265
                0108   C6  9F 001FA          PUSHAB  P.AEJ                           ; 7267
                       04  11 001FE          BRB     36$                             ;
                010E   C6  9F 00200 35$:     PUSHAB  P.AEK                           ; 7270
          67           01  FB 00204 36$:     CALLS   #1, DBG$PRINT                   ;
          05   02      A5  91 00207 37$:     CMPB    2(R5), #5                       ; 7281
                       07  12 0020B          BNEQ    38$                             ;
          50   011C    C6  9E 0020D          MOVAB   P.AEM, R0                       ; 7282
                       37  11 00212          BRB     43$                             ;
          06   02      A5  91 00214 38$:     CMPB    2(R5), #6                       ; 7283
                       07  12 00218          BNEQ    39$                             ;
          50   0123    C6  9E 0021A          MOVAB   P.AEN, R0                       ; 7284
                       2A  11 0021F          BRB     43$                             ;
          08   02      A5  91 00221 39$:     CMPB    2(R5), #8                       ; 7285
                       07  12 00225          BNEQ    40$                             ;
          50   012D    C6  9E 00227          MOVAB   P.AEO, R0                       ; 7286
                       1D  11 0022C          BRB     43$                             ;
          09   02      A5  91 0022E 40$:     CMPB    2(R5), #9                       ; 7287
                       07  12 00232          BNEQ    41$                             ;
          50   013A    C6  9E 00234          MOVAB   P.AEP, R0                       ; 7288
                       10  11 00239          BRB     43$                             ;
          07   02      A5  91 0023B 41$:     CMPB    2(R5), #7                       ; 7289
                       05  13 0023F          BEQL    42$                             ;
          7E           01  CE 00241          MNEGL   #1, -(SP)                       ;
                       07  11 00244          BRB     44$                             ;
          50   014A    C6  9E 00246 42$:     MOVAB   P.AEQ, R0                       ; 7290
                       50  DD 0024B 43$:     PUSHL   R0                              ;
                0114   C6  9F 0024D 44$:     PUSHAB  P.AEL                           ; 7278
    67                 02  FB 00251          CALLS   #2, DBG$PRINT                   ;
                       0D  11 00254          BRB     47$                             ; 7147
          02   01      A5  91 00256 45$:     CMPB    1(R5), #2                       ; 7299
                       07  12 0025A          BNEQ    47$                             ;
                0150   C6  9F 0025C          PUSHAB  P.AER                           ; 7300
    67                 01  FB 00260 46$:     CALLS   #1, DBG$PRINT                   ;
    07   03      A5  E9 00263 47$:     BLBC    3(R5), 48$                      ; 7307
```

```
                         015E   C6  9F 00267             PUSHAB  P.AES                                          ;
                     67          01  FB 0026B             CALLS   #1, DBG$PRINT                                 ;
                07   65          1A  E1 0026E 48$:        BBC     #26, (R5), 49$                                ; 7312
                         0168   C6  9F 00272             PUSHAB  P.AET                                         ;
                     67          01  FB 00276             CALLS   #1, DBG$PRINT                                 ;
                     01    01    A5  91 00279 49$:        CMPB    1(R5), #1                                     ; 7319
                                 43  12 0027D             BNEQ    53$                                           ;
                           28    A4  D5 0027F             TSTL    40(R4)                                        ; 7320
                                 3E  13 00282             BEQL    53$                                           ;
                     55    28    A4  D0 00284             MOVL    40(R4), OPCODE_LIST                           ; 7335
                     52          01  CE 00288             MNEGL   #1, COLUMN                                    ; 7336
                                 53  D4 0028B             CLRL    OPCODE                                        ; 7339
                29   65          53  E1 0028D 50$:        BBC     OPCODE, (OPCODE_LIST), 52$                    ;
                     50    01    A2  9E 00291             MOVAB   1(R2), R0                                     ; 7342
           52        50          03  EF 00295             EXTZV   #0, #3, R0, COLUMN                            ;
                                 03  12 0029A             BNEQ    51$                                           ; 7343
                     68          00  FB 0029C             CALLS   #0, DBG$NEWLINE                               ;
                     50 00000000G00043  3C 0029F 51$:      MOVZWL  DBG$OPCODE_KIND_TABLE[OPCODE], INDEX          ; 7344
                     50          0A  C4 002A7             MULL2   #10, R0                                       ; 7346
                        00000000G00040  9F 002AA             PUSHAB  DBG$OPCODE_NAME_TABLE[R0]                     ;
                                 06  DD 002B1             PUSHL   #6                                            ;
                         0175   C6  9F 002B3             PUSHAB  P.AEU                                         ; 7345
                     67          03  FB 002B7             CALLS   #3, DBG$PRINT                                 ; 7346
                CB   53 000001FF  8F  F3 002BA 52$:       AOBLEQ  #511, OPCODE, 50$                             ; 7337
                     68          00  FB 002C2 53$:       CALLS   #0, DBG$NEWLINE                               ; 7358
                     01    1C    A4  D1 002C5             CMPL    28(R4), #1                                    ; 7363
                                 0D  13 002C9             BEQL    54$                                           ;
                           1C    A4  DD 002CB             PUSHL   28(R4)                                        ; 7366
                         017B   C6  9F 002CE             PUSHAB  P.AEV                                         ;
                     67          02  FB 002D2             CALLS   #2, DBG$PRINT                                 ;
                     68          00  FB 002D5             CALLS   #0, DBG$NEWLINE                               ; 7367
                           20    A4  D5 002D8 54$:        TSTL    32(R4)                                        ; 7373
                                 1A  13 002DB             BEQL    55$                                           ;
                     50    20    A4  D0 002DD             MOVL    32(R4), WHEN_ENTRY                            ; 7379
                     50    0C    A0  D0 002E1             MOVL    12(WHEN_ENTRY), STRING                        ; 7380
                           02    A0  9F 002E5             PUSHAB  2(STRING)                                     ; 7381
                     7E          60  3C 002E8             MOVZWL  (STRING), -(SP)                               ;
                                 6E  D7 002EB             DECL    (SP)                                          ;
                         018A   C6  9F 002ED             PUSHAB  P.AEW                                         ;
                     67          03  FB 002F1             CALLS   #3, DBG$PRINT                                 ;
                     68          00  FB 002F4             CALLS   #0, DBG$NEWLINE                               ; 7382
                           24    A4  D5 002F7 55$:        TSTL    36(R4)                                        ; 7388
                                 1A  13 002FA             BEQL    56$                                           ;
                     50    24    A4  D0 002FC             MOVL    36(R4), DO_LIST_ENTRY                         ; 7394
                     50    0C    A0  D0 00300             MOVL    12(DO_LIST_ENTRY), STRING                     ; 7395
                           02    A0  9F 00304             PUSHAB  2(STRING)                                     ; 7396
                     7E          60  3C 00307             MOVZWL  (STRING), -(SP)                               ;
                                 6E  D7 0030A             DECL    (SP)                                          ;
                         0198   C6  9F 0030C             PUSHAB  P.AEX                                         ;
                     67          03  FB 00310             CALLS   #3, DBG$PRINT                                 ;
                     68          00  FB 00313             CALLS   #0, DBG$NEWLINE                               ; 7397
                                 04  00316 56$:           RET                                                   ; 7400

; Routine Size:  791 bytes,    Routine Base:  DBG$CODE + 29E1
```

```
;  7294          7401  1 GLOBAL ROUTINE DELETE_EVENT_ENTRY(EVENT_ENTRY): NOVALUE =
;  7295          7402  1 !
;  7296          7403  1 ! FUNCTION
;  7297          7404  1 !       This routine is called to delete an event entry from its queue
;  7298          7405  1 !       (the command queue and event queue if linked in), as well as
;  7299          7406  1 !       any value descriptor, WHEN, or DO entries.
;  7300          7407  1 !
;  7301          7408  1 ! INPUTS
;  7302          7409  1 !       EVENT_ENTRY :           Pointer to the event entry to be deleted.
;  7303          7410  1 !
;  7304          7411  1 ! OUTPUTS
;  7305          7412  1 !       None
;  7306          7413  1 !
;  7307          7414  1
;  7308          7415  2     BEGIN
;  7309          7416  2
;  7310          7417  2     MAP
;  7311          7418  2         EVENT_ENTRY :       REF EVENT$EVENT_DESCRIPTOR;
;  7312          7419  2
;  7313          7420  2     LOCAL
;  7314          7421  2         WHEN_ENTRY :        REF EVENT$WHEN_DESCRIPTOR,      ! WHEN pointer
;  7315          7422  2         DO_LIST_ENTRY :     REF EVENT$DO_LIST_DESCRIPTOR,   ! DO List pointer
;  7316          7423  2
;  7317          7424  2         ADDRESS,                                            ! Release address
;  7318          7425  2         SYMID_LIST,
;  7319          7426  2         DUMMY;
;  7320          7427  2
;  7321          7428  2
;  7322          7429  2     ! *****SSI
;  7323          7430  2     ! We just had Reserved Operand fault from RET of System service.
;  7324          7431  2     ! It is time to take the entry off, erase what we have done, just
;  7325          7432  2     ! as if nothing ever happened.
;  7326          7433  2     !
;  7327          7434  2     IF .EVENT_ENTRY [EVENT$B_CMD_TYPE] EQL EVENT$K_TYPE_SSINT
;  7328          7435  2     THEN
;  7329          7436  3         BEGIN
;  7330          7437  3         LOCAL
;  7331          7438  3             BIT_15_CNT: REF VECTOR[,BYTE],
;  7332          7439  3             BIT_15_CNT_SSV: REF VECTOR[,BYTE],
;  7333          7440  3             USERS_FP: REF BLOCK [,BYTE];
;  7334          7441  3
;  7335          7442  3         BIT_15_CNT_SSV = .EVENT_ENTRY [EVENT$L_SSV_COUNT];
;  7336          7443  3         BIT_15_CNT = .EVENT_ENTRY [EVENT$L_FP_RET_CNT];
;  7337          7444  3
;  7338          7445  3
;  7339          7446  3         ! We have got reserved operand fault.
;  7340          7447  3         !
;  7341          7448  3         BIT_15_CNT[0] = .BIT_15_CNT[0] - 1;
;  7342          7449  3         USERS_FP = .EVENT_ENTRY [EVENT$L_USERS_FP];
;  7343          7450  3
;  7344          7451  3
;  7345          7452  3         ! This service is leaving.
;  7346          7453  3         !
;  7347          7454  3         IF .BIT_15_CNT[0] EQL 0
;  7348          7455  3         THEN
;  7349          7456  4             BEGIN
;  7350          7457  4
```

```
;  7351     7458  4
;  7352     7459  4              ! Get ready to report watched value at T-bit RET time.
;  7353     7460  4              !
;  7354     7461  4              IF .DBG$GB_SET_SSI_CNT THEN SKIP_WATCHES = TRUE;
;  7355     7462  4              (USERS_FP [SF$W_SAVE_PSW])<15,1,0> = FALSE;
;  7356     7463  4 ! Since we only have one entry on the event queue, so there is no need
;  7357     7464  4 ! to match the nested system calls.
;  7358     7465  4 !            BIT_15_CNT_SSV[0] = .BIT_15_CNT_SSV[0] - 1;
;  7359     7466  4
;  7360     7467  4              ! Just about to return to the caller.
;  7361     7468  4              !
;  7362     7469  4              BIT_15_CNT_SSV[0] = 0;
;  7363     7470  4
;  7364     7471  4              ! Cycle is all done.
;  7365     7472  4              !
;  7366     7473  4              DBG$GB_SET_WATCH_FLAG = FALSE;
;  7367     7474  4              END
;  7368     7475  4
;  7369     7476  3          ELSE
;  7370     7477  3
;  7371     7478  3
;  7372     7479  3              ! Still more left to go.  More levels to go through, for example,
;  7373     7480  3              ! SDBG sees it, next IDBG should see it.  Since we only make the
;  7374     7481  3              ! reserved operand fault goes through one level, this should never
;  7375     7482  3              ! happen.
;  7376     7483  3              !
;  7377     7484  3              (USERS_FP [SF$W_SAVE_PSW])<15,1,0> = TRUE;
;  7378     7485  2          END;
;  7379     7486  2
;  7380     7487  2
;  7381     7488  2      ! If this entry is a skips-event, remove it from its parent's queue
;  7382     7489  2      ! first.  If its NOT a skips, and the parent queue is not empty, we
;  7383     7490  2      ! have (at the moment) and error...  The parent still has active
;  7384     7491  2      ! offspring.
;  7385     7492  2      !
;  7386     7493  2      IF .EVENT_ENTRY [EVENT$B_CMD_TYPE] EQLU EVENT$K_TYPE_SKIPS OR
;  7387     7494  2         .EVENT_ENTRY [EVENT$B_CMD_TYPE] EQLU EVENT$K_TYPE_IGNOR OR
;  7388     7495  2         .EVENT_ENTRY [EVENT$B_CMD_TYPE] EQLU EVENT$K_TYPE_SSINT
;  7389     7496  2      THEN
;  7390     7497  2          REMQUE1 (.EVENT_ENTRY)
;  7391     7498  2
;  7392     7499  2      ELSE IF .EVENT_ENTRY [EVENT$L_EXC_FLINK] NEQA .EVENT_ENTRY OR
;  7393     7500  2              .EVENT_ENTRY [EVENT$L_EXC_BLINK] NEQA .EVENT_ENTRY
;  7394     7501  2      THEN
;  7395     7502  2          RETURN;
;  7396     7503  2
;  7397     7504  2
;  7398     7505  2      ! Remove the entry from the command queue.
;  7399     7506  2      !
;  7400     7507  2      REMQUE (.EVENT_ENTRY, EVENT_ENTRY);
;  7401     7508  2
;  7402     7509  2
;  7403     7510  2      ! If this is not a STEPS, SKIPS or IGNOR entry....
;  7404     7511  2      !
;  7405     7512  2      IF (.EVENT_ENTRY [EVENT$B_CMD_TYPE] NEQU EVENT$K_TYPE_STEPS) AND
;  7406     7513  2         (.EVENT_ENTRY [EVENT$B_CMD_TYPE] NEQU EVENT$K_TYPE_SKIPS) AND
;  7407     7514  2         (.EVENT_ENTRY [EVENT$B_CMD_TYPE] NEQU EVENT$K_TYPE_IGNOR) AND
```

```
;  7408          7515  3        (.EVENT_ENTRY [EVENT$B_CMD_TYPE] NEQU EVENT$K_TYPE_SSINT)
;  7409          7516  2    THEN
;  7410          7517  2        BEGIN
;  7411          7518  2
;  7412          7519  2
;  7413          7520  2        ! If the entry is an access type, release any primary
;  7414          7521  2        ! and value descriptor.
;  7415          7522  2        !
;  7416          7523  2        IF .EVENT_ENTRY [EVENT$B_CMD_KIND] EQLU EVENT$K_KIND_ACC
;  7417          7524  3        THEN
;  7418          7525  4            BEGIN
;  7419          7526  4
;  7420          7527  4
;  7421          7528  4            ! Release any page entries.
;  7422          7529  4            !
;  7423          7530  4            IF .EVENT_ENTRY [EVENT$B_SUB_KIND] EQLU EVENT$K_ACC_MDFY
;  7424          7531  4            THEN
;  7425          7532  5                BEGIN
;  7426          7533  5                LOCAL
;  7427          7534  5                    PAGE_LIST : REF VECTOR [, LONG];
;  7428          7535  5
;  7429          7536  5
;  7430          7537  5                ! Get the pages associated with this value
;  7431          7538  5                !
;  7432          7539  5                IF DBG$NGET_PAGES (.EVENT_ENTRY [EVENT$L_PRIMARY],
;  7433          7540  5                                   PAGE_LIST,
;  7434          7541  5                                   DUMMY
;  7435          7542  5                                  )
;  7436          7543  5                THEN
;  7437          7544  6                    BEGIN
;  7438          7545  6
;  7439          7546  6
;  7440          7547  6                    ! While there are entries in the page list, delete a
;  7441          7548  6                    ! page reference from the page queue.
;  7442          7549  6                    !
;  7443          7550  6                    DO
;  7444          7551  7                        BEGIN
;  7445          7552  7                        LOCAL
;  7446          7553  7                            PAGE_ENTRY :    REF EVENT$PAGE_DESCRIPTOR;
;  7447          7554  7
;  7448          7555  7
;  7449          7556  7                        ! Search through the existing entries in the page
;  7450          7557  7                        ! queue for this page.
;  7451          7558  7                        !
;  7452          7559  7                        PAGE_ENTRY = .EVENT$PAGE_QUEUE [L_QUEUE_FLINK];
;  7453          7560  7                        WHILE .PAGE_ENTRY NEQA EVENT$PAGE_QUEUE DO
;  7454          7561  8                            BEGIN
;  7455          7562  8                            IF .PAGE_ENTRY [EVENT$L_PAGE_ADDRESS] EQLA
;  7456          7563  8                               .PAGE_LIST [1]
;  7457          7564  8                            THEN
;  7458          7565  8
;  7459          7566  8
;  7460          7567  8                                ! A match is found, so de-bump the
;  7461          7568  8                                ! reference count, deleting the entry if
;  7462          7569  8                                ! the count goes to zero.
;  7463          7570  8                                !
;  7464          7571  9                                BEGIN
```

```
;  7465     7572  9                        PAGE_ENTRY [EVENT$W_PAGE_REF_COUNT] =
;  7466     7573  9                            .PAGE_ENTRY [EVENT$W_PAGE_REF_COUNT] - 1;
;  7467     7574  9                        IF .PAGE_ENTRY [EVENT$W_PAGE_REF_COUNT] EQLU 0
;  7468     7575  9                        THEN
;  7469     7576 10                            BEGIN
;  7470     7577 10                            REMQUE (.PAGE_ENTRY, PAGE_ENTRY);
;  7471     7578 10                            DBG$REL_MEMORY (.PAGE_ENTRY);
;  7472     7579 10                            EXITLOOP;
;  7473     7580  9                            END;
;  7474     7581  9
;  7475     7582  8                        END;
;  7476     7583  8
;  7477     7584  8
;  7478     7585  8                        ! Not found yet, point to the next one.
;  7479     7586  8                        !
;  7480     7587  8                        PAGE_ENTRY = .PAGE_ENTRY [EVENT$L_PAGE_FLINK];
;  7481     7588  7                        END;
;  7482     7589  7
;  7483     7590  7                    PAGE_LIST = .PAGE_LIST [0];
;  7484     7591  7                    END
;  7485     7592  6                WHILE .PAGE_LIST;
;  7486     7593  5                END;
;  7487     7594  4            END;
;  7488     7595  4
;  7489     7596  4
;  7490     7597  4        ! Notify the RST to release symid's.
;  7491     7598  4        !
;  7492     7599  4        IF DBG$NGET_SYMID (.EVENT_ENTRY [EVENT$L_PRIMARY],
;  7493     7600  4                           SYMID_LIST,
;  7494     7601  4                           DUMMY
;  7495     7602  4                          )
;  7496     7603  4        THEN
;  7497     7604  4            DBG$STA_UNLOCK_SYMID (.SYMID_LIST);
;  7498     7605  4
;  7499     7606  4
;  7500     7607  4        ! Release the primary descriptor
;  7501     7608  4        !
;  7502     7609  4        DBG$NFREE_DESC (.EVENT_ENTRY [EVENT$L_PRIMARY], DUMMY);
;  7503     7610  4
;  7504     7611  4
;  7505     7612  4        ! Release any value descriptor.
;  7506     7613  4        !
;  7507     7614  4        IF .EVENT_ENTRY [EVENT$V_HAS_VAL_DSCR]
;  7508     7615  4        THEN
;  7509     7616  5            BEGIN
;  7510     7617  5
;  7511     7618  5
;  7512     7619  5            ! Notify the RST to release symid's.
;  7513     7620  5            !
;  7514     7621  5            IF DBG$NGET_SYMID (.EVENT_ENTRY [EVENT$L_VALDESC],
;  7515     7622  5                               SYMID_LIST,
;  7516     7623  5                               DUMMY
;  7517     7624  5                              )
;  7518     7625  5            THEN
;  7519     7626  5                DBG$STA_UNLOCK_SYMID (.SYMID_LIST);
;  7520     7627  5
;  7521     7628  5
```

```
;  7522     7629  5                ! Release the value descriptor
;  7523     7630  5                !
;  7524     7631  5                DBG$NFREE_DESC (.EVENT_ENTRY [EVENT$L_VALDESC], DUMMY);
;  7525     7632  5                END;
;  7526     7633  4            END;
;  7527     7634  4
;  7528     7635  4
;  7529     7636  4        ! Release any WHEN Entries.
;  7530     7637  4        !
;  7531     7638  4        WHEN_ENTRY = .EVENT_ENTRY [EVENT$L_WHEN];
;  7532     7639  4        IF .WHEN_ENTRY NEQA 0
;  7533     7640  4        THEN
;  7534     7641  4            BEGIN
;  7535     7642  4
;  7536     7643  4
;  7537     7644  4            ! We have a WHEN entry.  Decrement its reference count.
;  7538     7645  4            !
;  7539     7646  4            WHEN_ENTRY [EVENT$L_WHEN_COUNT] =
;  7540     7647  4                .WHEN_ENTRY [EVENT$L_WHEN_COUNT] - 1;
;  7541     7648  4
;  7542     7649  4
;  7543     7650  4            ! Delete it if the count goes to 0 (or less !!!).
;  7544     7651  4            !
;  7545     7652  4            IF .WHEN_ENTRY [EVENT$L_WHEN_COUNT] LEQU 0
;  7546     7653  4            THEN
;  7547     7654  5                BEGIN
;  7548     7655  5
;  7549     7656  5
;  7550     7657  5                ! Release the expression buffer.
;  7551     7658  5                !
;  7552     7659  5                DBG$REL_MEMORY (.WHEN_ENTRY [EVENT$L_WHEN_POINT]);
;  7553     7660  5
;  7554     7661  5
;  7555     7662  5                ! Release the WHEN descriptor.
;  7556     7663  5                !
;  7557     7664  5                DBG$REL_MEMORY (.WHEN_ENTRY);
;  7558     7665  5                END
;  7559     7666  4            END;
;  7560     7667  4
;  7561     7668  4
;  7562     7669  4        ! Release any DO List Entries.
;  7563     7670  4        !
;  7564     7671  4        DO_LIST_ENTRY = .EVENT_ENTRY [EVENT$L_DO];
;  7565     7672  4        IF .DO_LIST_ENTRY NEQA 0
;  7566     7673  4        THEN
;  7567     7674  4            BEGIN
;  7568     7675  4
;  7569     7676  4
;  7570     7677  4            ! We have a DO List entry.  Decrement its reference count.
;  7571     7678  4            !
;  7572     7679  4            DO_LIST_ENTRY [EVENT$L_DO_LIST_COUNT] =
;  7573     7680  4                .DO_LIST_ENTRY [EVENT$L_DO_LIST_COUNT] - 1;
;  7574     7681  4
;  7575     7682  4
;  7576     7683  4            ! Delete it if the count goes to 0 (or less !!!).
;  7577     7684  4            !
;  7578     7685  4            IF .DO_LIST_ENTRY [EVENT$L_DO_LIST_COUNT] LEQU 0
```

```
;  7579     7686  4              THEN
;  7580     7687  5                  BEGIN
;  7581     7688  5          !       LOCAL
;  7582     7689  5          !           PREV_ENTRY: REF EVENT$DO_LIST_DESCRIPTOR,
;  7583     7690  5          !           NEXT_ENTRY: REF EVENT$DO_LIST_DESCRIPTOR;
;  7584     7691  5
;  7585     7692  5
;  7586     7693  5                  ! Release the command buffer.
;  7587     7694  5                  !
;  7588     7695  5                  DBG$REL_MEMORY (.DO_LIST_ENTRY [EVENT$L_DO_LIST_POINT]);
;  7589     7696  5
;  7590     7697  5
;  7591     7698  5                  ! Unlink this DO_LIST entry from the list. The REMQUE
;  7592     7699  5                  ! instruction does the same as the following four lines
;  7593     7700  5                  ! (I hope).
;  7594     7701  5                  !
;  7595     7702  5                  PREV_ENTRY = .DO_LIST_ENTRY [EVENT$L_DO_LIST_BLINK];
;  7596     7703  5                  NEXT_ENTRY = .DO_LIST_ENTRY [EVENT$L_DO_LIST_FLINK];
;  7597     7704  5                  PREV_ENTRY [EVENT$L_DO_LIST_FLINK] = .NEXT_ENTRY;
;  7598     7705  5                  NEXT_ENTRY [EVENT$L_DO_LIST_BLINK] = .PREV_ENTRY;
;  7599     7706  5                  !
;  7600     7707  5                  REMQUE (.DO_LIST_ENTRY, DO_LIST_ENTRY);
;  7601     7708  5
;  7602     7709  5
;  7603     7710  5                  ! Release the DO_LIST descriptor.
;  7604     7711  5                  !
;  7605     7712  5                  DBG$REL_MEMORY (.DO_LIST_ENTRY);
;  7606     7713  5                  END;
;  7607     7714  4              END;
;  7608     7715  3          END;
;  7609     7716  2
;  7610     7717  2
;  7611     7718  2          ! Release the opcode list.
;  7612     7719  2          !
;  7613     7720  2          IF (.EVENT_ENTRY [EVENT$B_CMD_KIND] EQLU EVENT$K_KIND_INS) AND
;  7614     7721  2             (.EVENT_ENTRY [EVENT$L_OPCODE_LIST] NEQA 0) AND
;  7615     7722  2             (.EVENT_ENTRY [EVENT$L_OPCODE_LIST] NEQA DBG$OPCODES_CALL) AND
;  7616     7723  3             (.EVENT_ENTRY [EVENT$L_OPCODE_LIST] NEQA DBG$OPCODES_BRANCH)
;  7617     7724  2          THEN
;  7618     7725  2              DBG$REL_MEMORY (.EVENT_ENTRY [EVENT$L_OPCODE_LIST]);
;  7619     7726  2
;  7620     7727  2
;  7621     7728  2          ! Release Copied FP.
;  7622     7729  2          !
;  7623     7730  2          IF .EVENT_ENTRY [EVENT$B_CMD_TYPE] EQLU EVENT$K_TYPE_SKIPS OR
;  7624     7731  2             .EVENT_ENTRY [EVENT$B_CMD_TYPE] EQLU EVENT$K_TYPE_STEPS
;  7625     7732  2          THEN
;  7626     7733  2              IF .EVENT_ENTRY [EVENT$L_CALL_FRAME] NEQA 0
;  7627     7734  2              THEN
;  7628     7735  2                  DBG$REL_MEMORY (.EVENT_ENTRY [EVENT$L_CALL_FRAME]);
;  7629     7736  2
;  7630     7737  2
;  7631     7738  2          ! Finally, release the Event Entry, itself.
;  7632     7739  2          !
;  7633     7740  2          DBG$REL_MEMORY (.EVENT_ENTRY);
;  7634     7741  1          END;
```

```
                                07FC 00000          .ENTRY  DELETE_EVENT_ENTRY, Save R2,R3,R4,R5,R6,R7,-;  7401
                                                            R8,R9,R10
                5A 00000000G    00   9E 00002       MOVAB   DBG$NFREE_DESC, R10                            :
                59 00000000G    00   9E 00009       MOVAB   DBG$STA_UNLOCK_SYMID, R9                       :
                58 00000000G    00   9E 00010       MOVAB   DBG$NGET_SYMID, R8                             :
                57 00000000'    EF   9E 00017       MOVAB   EVENT$PAGE_QUEUE, R7                           :
                56 00000000'    EF   9E 0001E       MOVAB   SKIP_WATCHES, R6                               :
                55 00000000G    00   9E 00025       MOVAB   DBG$REL_MEMORY, R5                             :
                5E              0C   C2 0002C       SUBL2   #12, SP                                        :
                52        04    AC   D0 0002F       MOVL    EVENT_ENTRY, R2                                :  7434
                                54   D4 00033       CLRL    R4                                             :
                06        14    A2   91 00035       CMPB    20(R2), #6                                     :
                                2F   12 00039       BNEQ    3$                                             :
                                54   D6 0003B       INCL    R4                                             :
                53        1C    A2   D0 0003D       MOVL    28(R2), BIT_15_CNT_SSV                         :  7442
                51        2C    A2   D0 00041       MOVL    44(R2), BIT_15_CNT                             :  7443
                                61   97 00045       DECB    (BIT_15_CNT)                                   :  7448
                50        24    A2   D0 00047       MOVL    36(R2), USERS_FP                               :  7449
                                61   95 0004B       TSTB    (BIT_15_CNT)                                   :  7454
                                16   12 0004D       BNEQ    2$                                             :
                03 00000000G    00   E9 0004F       BLBC    DBG$GB_SET_SSI_CNT, 1$                         :  7461
                66              01   D0 00056       MOVL    #1, SKIP_WATCHES                               :
    05          A0        80    8F   8A 00059 1$:   BICB2   #128, 5(USERS_FP)                             :  7462
                                63   94 0005E       CLRB    (BIT_15_CNT_SSV)                               :  7469
                          CD    A7   94 00060       CLRB    DBG$GB_SET_WATCH_FLAG                          :  7473
                                05   11 00063       BRB     3$                                             :  7454
    05          A0        80    8F   88 00065 2$:   BISB2   #128, 5(USERS_FP)                             :  7484
                04        14    A2   91 0006A 3$:   CMPB    20(R2), #4                                     :  7493
                                09   13 0006E       BEQL    4$                                             :
                05        14    A2   91 00070       CMPB    20(R2), #5                                     :  7494
                                03   13 00074       BEQL    4$                                             :
                09                   54   F9 00076       BLBC    R4, 5$                                     :  7495
                                     52   DD 00079 4$:   PUSHL   R2                                         :  7497
            FB86    CF               01   FB 0007B       CALLS   #1, REMQUE1                                :
                                     0D   11 00080       BRB     7$                                         :
                52        08    A2   D1 00082 5$:   CMPL    8(R2), R2                                      :  7499
                                04   12 00086       BNEQ    6$                                             :
                52        0C    A2   D1 00088       CMPL    12(R2), R2                                     :  7500
                                01   13 0008C 6$:   BEQL    7$                                             :
                                     04 0008E       RET                                                    :
            04  AC                  62   0F 0008F 7$:   REMQUE  (R2), EVENT_ENTRY                          :  7507
                50        04    AC   D0 00093       MOVL    EVENT_ENTRY, R0                                :  7512
                03        14    A0   91 00097       CMPB    20(R0), #3                                     :
                                10   13 0009B       BEQL    8$                                             :
                04        14    A0   91 0009D       CMPB    20(R0), #4                                     :  7513
                                0A   13 000A1       BEQL    8$                                             :
                05        14    A0   91 000A3       CMPB    20(R0), #5                                     :  7514
                                04   13 000A7       BEQL    8$                                             :
                06        14    A0   91 000A9       CMPB    20(R0), #6                                     :  7515
                                03   12 000AD 8$:   BNEQ    9$                                             :
                              00C7   31 000AF       BRW     20$                                            :
                          15    A0   95 000B2 9$:   TSTB    21(R0)                                         :  7523
                                03   13 000B5       BEQL    10$                                            :
```

```
                                   008C   31 000B7          BRW      18$                                      ;
                02          16       A0   91 000BA  10$:    CMPB     22(R0), #2                               ; 7530
                                     41   12 000BE          BNEQ     15$                                      ;
                            08       AE   9F 000C0          PUSHAB   DUMMY                                    ; 7539
                            04       AE   9F 000C3          PUSHAB   PAGE_LIST                                ;
                            28       A0   DD 000C6          PUSHL    40(R0)                                   ;
        00000000G  00                03   FB 000C9          CALLS    #3, DBG$NGET_PAGES                       ;
                   2E                50   E9 000D0          BLBC     R0, 15$                                  ;
                   52                67   D0 000D3  11$:    MOVL     EVENT$PAGE_QUEUE, PAGE_ENTRY             ; 7559
                   50                67   9E 000D6  12$:    MOVAB    EVENT$PAGE_QUEUE, R0                     ; 7560
                   50                52   D1 000D9          CMPL     PAGE_ENTRY, R0                           ;
                                     1E   13 000DC          BEQL     14$                                      ;
                   50                6E   D0 000DE          MOVL     PAGE_LIST, R0                            ; 7563
             04    A0       08       A2   D1 000E1          CMPL     8(PAGE_ENTRY), 4(R0)                     ;
                                     0F   12 000E6          BNEQ     13$                                      ;
                            0C       A2   B7 000E8          DECW     12(PAGE_ENTRY)                           ; 7573
                                     0A   12 000EB          BNEQ     13$                                      ; 7574
                   52                62   0F 000ED          REMQUE   (PAGE_ENTRY), PAGE_ENTRY                 ; 7577
                                     52   DD 000F0          PUSHL    PAGE_ENTRY                               ; 7578
                   65                01   FB 000F2          CALLS    #1, DBG$REL_MEMORY                       ;
                                     05   11 000F5          BRB      14$                                      ; 7576
                   52                62   D0 000F7  13$:    MOVL     (PAGE_ENTRY), PAGE_ENTRY                 ; 7587
                                     DA   11 000FA          BRB      12$                                      ; 7560
                                     9E   DD 000FC  14$:    PUSHL    @PAGE_LIST                               ; 7590
                   D2                6E   E8 000FE          BLBS     PAGE_LIST, 11$                           ; 7592
                            08       AE   9F 00101  15$:    PUSHAB   DUMMY                                    ; 7599
                            08       AE   9F 00104          PUSHAB   SYMID_LIST                               ;
                   52       04       AC   D0 00107          MOVL     EVENT_ENTRY, R2                          ;
                            28       A2   DD 0010B          PUSHL    40(R2)                                   ;
                   68                03   FB 0010E          CALLS    #3, DBG$NGET_SYMID                       ;
                   06                50   E9 00111          BLBC     R0, 16$                                  ;
                            04       AE   DD 00114          PUSHL    SYMID_LIST                               ; 7604
                   69                01   FB 00117          CALLS    #1, DBG$STA_UNLOCK_SYMID                 ;
                            08       AE   9F 0011A  16$:    PUSHAB   DUMMY                                    ; 7609
                            28       A2   DD 0011D          PUSHL    40(R2)                                   ;
                   6A                02   FB 00120          CALLS    #2, DBG$NFREE_DESC                       ;
       1E          17       A2       03   E1 00123          BBC      #3, 23(R2), 18$                          ; 7614
                            08       AE   9F 00128          PUSHAB   DUMMY                                    ; 7621
                            08       AE   9F 0012B          PUSHAB   SYMID_LIST                               ;
                            30       A2   DD 0012E          PUSHL    48(R2)                                   ;
                   68                03   FB 00131          CALLS    #3, DBG$NGET_SYMID                       ;
                   06                50   E9 00134          BLBC     R0, 17$                                  ;
                            04       AE   DD 00137          PUSHL    SYMID_LIST                               ; 7626
                   69                01   FB 0013A          CALLS    #1, DBG$STA_UNLOCK_SYMID                 ;
                            08       AE   9F 0013D  17$:    PUSHAB   DUMMY                                    ; 7631
                            30       A2   DD 00140          PUSHL    48(R2)                                   ;
                   6A                02   FB 00143          CALLS    #2, DBG$NFREE_DESC                       ;
                   53       04       AC   D0 00146  18$:    MOVL     EVENT_ENTRY, R3                          ; 7638
                   52       20       A3   D0 0014A          MOVL     32(R3), WHEN_ENTRY                       ;
                                     10   13 0014E          BEQL     19$                                      ; 7639
                            08       A2   D7 00150          DECL     8(WHEN_ENTRY)                            ; 7647
                                     0B   12 00153          BNEQ     19$                                      ; 7652
                            0C       A2   DD 00155          PUSHL    12(WHEN_ENTRY)                           ; 7659
                   65                01   FB 00158          CALLS    #1, DBG$REL_MEMORY                       ;
                                     52   DD 0015B          PUSHL    WHEN_ENTRY                               ; 7664
                   65                01   FB 0015D          CALLS    #1, DBG$REL_MEMORY                       ;
                   52       24       A3   D0 00160  19$:    MOVL     36(R3), DO_LIST_ENTRY                    ; 7671
```

```
                         13  13 00164          BEQL    20$                                     ; 7672
                      08 A2  D7 00166          DECL    8(DO_LIST_ENTRY)                        ; 7680
                         0E  12 00169          BNEQ    20$                                     ; 7685
                      0C A2  DD 00168          PUSHL   12(DO_LIST_ENTRY)                       ; 7695
                   65    01  FB 0016E          CALLS   #1, DBG$REL_MEMORY
                   52       62 0F 00171       REMQUE  (DO_LIST_ENTRY), DO_LIST_ENTRY          ; 7707
                            52 DD 00174       PUSHL   DO_LIST_ENTRY                           ; 7712
                   65    01  FB 00176          CALLS   #1, DBG$REL_MEMORY
                   52    04  AC  D0 00179 20$: MOVL    EVENT_ENTRY, R2                         ; 7720
                   01    15  A2  91 0017D      CMPB    21(R2), #1
                         21  12 00181          BNEQ    21$
                      28 A2  D5 00183          TSTL    40(R2)                                  ; 7721
                         1C  13 00186          BEQL    21$
                   50 0088  C6  9E 00188       MOVAB   DBG$OPCODES_CALL, R0                    ; 7722
                   50    28 A2  D1 0018D      CMPL    40(R2), R0
                         11  13 00191          BEQL    21$
                   50 00C8  C6  9E 00193       MOVAB   DBG$OPCODES_BRANCH, R0                  ; 7723
                   50    28 A2  D1 00198      CMPL    40(R2), R0
                         06  13 0019C          BEQL    21$
                      28 A2  DD 0019E          PUSHL   40(R2)                                  ; 7725
                   65    01  FB 001A1          CALLS   #1, DBG$REL_MEMORY
                   04    14  A2  91 001A4 21$: CMPB    20(R2), #4                              ; 7730
                         06  13 001A8          BEQL    22$
                   03    14  A2  91 001AA      CMPB    20(R2), #3                              ; 7731
                         0B  12 001AE          BNEQ    23$
                      3C A2  D5 001B0 22$:     TSTL    60(R2)                                  ; 7733
                         06  13 001B3          BEQL    23$
                      3C A2  DD 001B5          PUSHL   60(R2)                                  ; 7735
                   65    01  FB 001B8          CALLS   #1, DBG$REL_MEMORY
                         52  DD 001BB 23$:     PUSHL   R2                                      ; 7740
                   65    01  FB 001BD          CALLS   #1, DBG$REL_MEMORY
                         04  001C0            RET                                             ; 7741

; Routine Size:  449 bytes,    Routine Base:  DBG$CODE + 2CF8
```

```
;  7636          7742  1 ROUTINE COMPARE_VMSDESC(DESC1, DESC2) =
;  7637          7743  1 !
;  7638          7744  1 ! FUNCTION
;  7639          7745  1 !       Determines whether two VMS descriptors represent the same area
;  7640          7746  1 !       of storage. This is used, for example, in processing the SET WATCH
;  7641          7747  1 !       command, to determine whether we already have a watchpoint on
;  7642          7748  1 !       the given data item.
;  7643          7749  1 !
;  7644          7750  1 ! INPUTS
;  7645          7751  1 !       DESC1 -         Points to a VMS descriptor
;  7646          7752  1 !       DESC2 -         Points to a VMS descriptor
;  7647          7753  1 !
;  7648          7754  1 ! OUTPUTS
;  7649          7755  1 !       The value TRUE or FALSE is returned.
;  7650          7756  1 !
;  7651          7757  2     BEGIN
;  7652          7758  2     MAP
;  7653          7759  2         DESC1: REF DBG$STG_DESC,
;  7654          7760  2         DESC2: REF DBG$STG_DESC;
;  7655          7761  2
;  7656          7762  2     LOCAL
;  7657          7763  2         LENGTH1,
;  7658          7764  2         LENGTH2,
;  7659          7765  2         OFFSET1,
;  7660          7766  2         OFFSET2;
;  7661          7767  2
;  7662          7768  2     ! Check whether the data given by the descriptors have
;  7663          7769  2     ! the same address.
;  7664          7770  2     !
;  7665          7771  2
;  7666          7772  2     IF .DESC1[DSC$B_CLASS] EQL DSC$K_CLASS_UBS
;  7667          7773  2     THEN OFFSET1 = .DESC1[DSC$L_POS] ELSE OFFSET1 = 0;
;  7668          7774  2
;  7669          7775  2     IF .DESC2[DSC$B_CLASS] EQL DSC$K_CLASS_UBS
;  7670          7776  2     THEN OFFSET2 = .DESC2[DSC$L_POS] ELSE OFFSET2 = 0;
;  7671          7777  2
;  7672          7778  2     IF (.DESC1[DSC$A_POINTER] + .OFFSET1<3,29,1>) NEQ
;  7673          7779  2        (.DESC2[DSC$A_POINTER] + .OFFSET2<3,29,1>)THEN RETURN FALSE;
;  7674          7780  2
;  7675          7781  2     IF .OFFSET1<0,3,0> NEQ .OFFSET2<0,3,0> THEN RETURN FALSE;
;  7676          7782  2
;  7677          7783  2     ! Compare the lengths.
;  7678          7784  2     ! (Do we really want to do this? The implications are:
;  7679          7785  2     !
;  7680          7786  2     ! SET WATCH ARRAY and SET WATCH ARRAY(1) are two different watchpoints,
;  7681          7787  2     ! since the watched areas have different lengths. This behavior seems
;  7682          7788  2     ! to be correct.
;  7683          7789  2     !
;  7684          7790  2     ! Say X is a byte variable at address 200. Say the default type is LONG.
;  7685          7791  2     ! Then SET WATCH X and SET WATCH 200 will be treated as two different
;  7686          7792  2     ! watchpoints (since one has length 1 and the other has length 4).
;  7687          7793  2     ! This does not seem to be a good thing. This problem could be solved
;  7688          7794  2     ! if we attempted to symbolize the "200" before setting the watchpoint
;  7689          7795  2     ! on it.)
;  7690          7796  2     !
;  7691          7797  2     LENGTH1 = DBG$DATA_LENGTH(.DESC1);
;  7692          7798  2     LENGTH2 = DBG$DATA_LENGTH(.DESC2);
```

```
; 7693         7799  2      RETURN .LENGTH1 EQL .LENGTH2;
; 7694         7800  1      END;



                                          007C 00000 COMPARE_VMSDESC:
                                                             .WORD   Save R2,R3,R4,R5,R6                      ; 7742
                     56 00000000G  00  9E 00002              MOVAB   DBG$DATA_LENGTH, R6                      ;
                     51        04  AC  D0 00009              MOVL    DESC1, R1                                ; 7772
                     0D        03  A1  91 0000D              CMPB    3(R1), #13                               ;
                                   06  12 00011              BNEQ    1$                                       ;
                     55        08  A1  D0 00013              MOVL    8(R1), OFFSET1                           ; 7773
                                   02  11 00017              BRB     2$                                       ;
                                   55  D4 00019 1$:          CLRL    OFFSET1                                  ;
                     52        08  AC  D0 0001B 2$:          MOVL    DESC2, R2                                ; 7775
                     0D        03  A2  91 0001F              CMPB    3(R2), #13                               ;
                                   06  12 00023              BNEQ    3$                                       ;
                     54        08  A2  D0 00025              MOVL    8(R2), OFFSET2                           ; 7776
                                   02  11 00029              BRB     4$                                       ;
                                   54  D4 0002B 3$:          CLRL    OFFSET2                                  ;
          53         55        FD  8F  78 0002D 4$:          ASHL    #-3, OFFSET1, R3                         ; 7778
                     53        04  A1  C0 00032              ADDL2   4(R1), R3                                ;
          50         54        FD  8F  78 00036              ASHL    #-3, OFFSET2, R0                         ; 7779
                     50        04  A2  C0 0003B              ADDL2   4(R2), R0                                ;
                     50            53  D1 0003F              CMPL    R3, R0                                   ;
                                   23  12 00042              BNEQ    6$                                       ;
          50         54            55  8D 00044              XORB3   OFFSET1, OFFSET2, R0                     ; 7781
                     07            50  93 00048              BITB    R0, #7                                   ;
                                   1A  12 0004B              BNEQ    6$                                       ;
                                   51  DD 0004D              PUSHL   R1                                       ; 7797
                     66            01  FB 0004F              CALLS   #1, DBG$DATA_LENGTH                      ;
                     53            50  D0 00052              MOVL    R0, LENGTH1                              ;
                                   52  DD 00055              PUSHL   R2                                       ; 7798
                     66            01  FB 00057              CALLS   #1, DBG$DATA_LENGTH                      ;
                                   51  D4 0005A              CLRL    R1                                       ; 7799
                     50            53  D1 0005C              CMPL    LENGTH1, LENGTH2                         ;
                                   02  12 0005F              BNEQ    5$                                       ;
                                   51  D6 00061              INCL    R1                                       ;
                     50            51  D0 00063 5$:          MOVL    R1, R0                                   ;
                                       04 00066              RET                                              ;
                               50  D4 00067 6$:              CLRL    R0                                       ; 7800
                                       04 00069              RET                                              ;

; Routine Size:  106 bytes,    Routine Base:  DBG$CODE + 2EB9


; 7695         7801  1
```

```
;  7697          7802  1 ROUTINE GET_WATCH_POINT_VALUE (EVENT : REF EVENT$EVENT_DESCRIPTOR): NOVALUE =
;  7698          7803  1 !
;  7699          7804  1 ! FUNCTION
;  7700          7805  1 !
;  7701          7806  1 !       This routine gets the new value for a variable that is watched.
;  7702          7807  1 !       It does so by converting the Primary in the Event Entry to a
;  7703          7808  1 !       Value Descriptor which contains the value.
;  7704          7809  1 !
;  7705          7810  1 !       Along the way, this routine checks that the Primary is still
;  7706          7811  1 !       valid (i.e., still points to the same storage that it did when
;  7707          7812  1 !       the watchpoint was set). If it is not valid, the Primary is
;  7708          7813  1 !       replaced by a Volatile Value Descriptor, and then the watchpoint
;  7709          7814  1 !       is on an absolute address instead of a primary expression.
;  7710          7815  1 !
;  7711          7816  1 ! INPUTS
;  7712          7817  1 !       EVENT   - Event entry pointer which describes the watchpoint.
;  7713          7818  1 !
;  7714          7819  1 ! OUTPUTS
;  7715          7820  1 !       The own variable NEWVALUE is set by this routine. The value in
;  7716          7821  1 !       NEWVALUE is then compared to the value in EVENT[EVENT$L_VALDESC]
;  7717          7822  1 !       to see if the watched variable has changed (this comparison
;  7718          7823  1 !       is done in DBG$EXCEPTION_HANDLER, the called of this routine).
;  7719          7824  1 !
;  7720          7825  2     BEGIN
;  7721          7826  2     LOCAL
;  7722          7827  2         DUMMY,                          ! Output parameter for subroutines
;  7723          7828  2         PRIMARY,                        ! Temporary copy of the Primary
;  7724          7829  2         SYMID_LIST,                     ! Symid list for LOCK_SYMID
;  7725          7830  2         VALDESC: REF DBG$VALDESC,       ! Value descriptor with value of Primary
;  7726          7831  2         V_VALDESC: REF DBG$VALDESC;     ! Volatile Value Descriptor
;  7727          7832  2
;  7728          7833  2
;  7729          7834  2     ! Copy the Primary since PRIM_TO_VAL sometimes has the side effect
;  7730          7835  2     ! of modifying its argument. This copy is done into temporary
;  7731          7836  2     ! memory (fourth parameter is FALSE). Then call PRIM_TO_VAL
;  7732          7837  2     ! and get a Volatile Value Descriptor (this will have
;  7733          7838  2     ! the real address of the watched variable embedded in it).
;  7734          7839  2     !
;  7735          7840  2     DBG$NCOPY_DESC (.EVENT[EVENT$L_PRIMARY], PRIMARY, DUMMY, FALSE);
;  7736          7841  2     INVALID_FLAG = FALSE;
;  7737          7842  2     PRIM_TO_VAL (.PRIMARY, DBG$K_V_VALUE_DESC, V_VALDESC);
;  7738          7843  2
;  7739          7844  2
;  7740          7845  2     ! Check whether the Primary is still valid, i.e., whether it still
;  7741          7846  2     ! points to the same storage as it did when the watchpoint was set.
;  7742          7847  2     !
;  7743          7848  2     IF .INVALID_FLAG OR
;  7744          7849  2        CH$NEQ (12, V_VALDESC[DBG$A_VALUE_VMSDESC], 12, EVENT[EVENT$A_VMSDESC])
;  7745          7850  2     THEN
;  7746          7851  3         BEGIN
;  7747          7852  3
;  7748          7853  3
;  7749          7854  3         ! Print an informational saying the Primary is no longer valid.
;  7750          7855  3         !
;  7751          7856  3         DBG$PRINT ($ASC ('%DEBUG-I-WATCHVAR, watched variable '));
;  7752          7857  3         DBG$PRINT_IDENTIFIER(.EVENT[EVENT$L_PRIMARY]);
;  7753          7858  3         IF .INVALID_FLAG
```

```
;  7754     7859  3          THEN
;  7755     7860  3              DBG$PRINT ($AC (' has gone out of scope'))
;  7756     7861  3          ELSE
;  7757     7862  3              DBG$PRINT ($AC (' now points to a different address'));
;  7758     7863  3          DBG$NEWLINE();
;  7759     7864  3
;  7760     7865  3
;  7761     7866  3          ! Free up the old Primary.
;  7762     7867  3          !
;  7763     7868  3          IF DBG$NGET_SYMID (.EVENT[EVENT$L_PRIMARY], SYMID_LIST, DUMMY)
;  7764     7869  3          THEN
;  7765     7870  3              DBG$STA_UNLOCK_SYMID (.SYMID_LIST);
;  7766     7871  3          DBG$NFREE_DESC (.EVENT[EVENT$L_PRIMARY], DUMMY);
;  7767     7872  3
;  7768     7873  3
;  7769     7874  3          ! Replace it with the Volatile Value Descriptor.
;  7770     7875  3          !
;  7771     7876  3          V_VALDESC = DBG$MAKE_VAL_DESC(EVENT[EVENT$A_VMSDESC], DBG$K_V_VALUE_DESC);
;  7772     7877  3          V_VALDESC[DBG$B_DHDR_LANG] = .DBG$GB_LANGUAGE;
;  7773     7878  3          IF DBG$NGET_SYMID (.V_VALDESC, SYMID_LIST, DUMMY)
;  7774     7879  3          THEN
;  7775     7880  3              DBG$STA_LOCK_SYMID (.SYMID_LIST);
;  7776     7881  3          DBG$NCOPY_DESC (.V_VALDESC, EVENT[EVENT$L_PRIMARY], DUMMY, TRUE);
;  7777     7882  3          END;
;  7778     7883  3
;  7779     7884  3
;  7780     7885  2      ! Build a Value Descriptor.
;  7781     7886  2      ! Then copy the value descriptor into permanent memory.
;  7782     7887  2      ! This is needed so that the storage stays around even if we
;  7783     7888  2      ! execute DO clauses on the watchpoint (which may in turn
;  7784     7889  2      ! cause calls to REL_TEMPMEM).
;  7785     7890  2      ! The memory for NEWVALUE is freed up in DBG$EXCEPTION_HANDLER
;  7786     7891  2      ! when it is finished with the Value Descriptor.
;  7787     7892  2      !
;  7788     7893  2      DBG$PRIM_TO_VAL(.V_VALDESC, DBG$K_VALUE_DESC, VALDESC);
;  7789     7894  2      DBG$NCOPY_DESC (.VALDESC, NEWVALUE, DUMMY, TRUE);
;  7790     7895  2      RETURN 0;
;  7791     7896  1      END;


                                                                    .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

48 43 54 41 57 2D 49 2D 47 55 42 45 44 25 24  0054B P.AEY:   .ASCII  \$%DEBUG-I-WATCHVAR, watched variable \      ;
61 76 20 64 65 68 63 74 61 77 20 2C 52 41 56  0055A                                                            ;
                        20 65 6C 62 61 69 72  00569                                                            ;
20 74 75 6F 20 65 6E 6F 67 20 73 61 68 20 16  00570 P.AEZ:   .ASCII  <22>\ has gone out of scope\               ;
                        65 70 6F 63 73 20 66 6F  0057F                                                          ;
6F 74 20 73 74 6E 69 6F 70 20 77 6F 6E 20 22  00587 P.AFA:   .ASCII  \" now points to a different address\      ;
64 61 20 74 6E 65 72 65 66 66 69 64 20 61 20  00596                                                            ;
                                  73 73 65 72 64  005A5                                                        ;


                                                                    .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                                    03FC 00000 GET_WATCH_POINT_VALUE:
```

```
                                59 00000000G   00    9E 00002          MOVAB   DBG$NGET_SYMID, R9
                                58 00000000G   00    9E 00009          MOVAB   DBG$PRINT, R8
                                57 00000000G   00    9E 00010          MOVAB   DBG$NCOPY_DESC, R7
                                56 00000000'   EF    9E 00017          MOVAB   P.AEY, R6
                                55 00000000'   EF    9E 0001E          MOVAB   INVALID_FLAG, R5
                                5E             14    C2 00025          SUBL2   #20, SP
                                               7E    D4 00028          CLRL    -(SP)                                   ;7840
                                         14    AE    9F 0002A          PUSHAB  DUMMY
                                         08    AE    9F 0002D          PUSHAB  PRIMARY
                                54       04    AC    D0 00030          MOVL    EVENT, R4
                                         28    A4    DD 00034          PUSHL   40(R4)
                                67             04    FB 00037          CALLS   #4, DBG$NCOPY_DESC
                                               65    D4 0003A          CLRL    INVALID_FLAG                            ;7841
                                         04    AE    9F 0003C          PUSHAB  V_VALDESC                               ;7842
                                7E       83    8F    9A 0003F          MOVZBL  #131, -(SP)
                                         08    AE    DD 00043          PUSHL   PRIMARY
                          0000V  CF              03    FB 00046          CALLS   #3, PRIM_TO_VAL
                                0F             65    E8 0004B          BLBS    INVALID_FLAG, 1$                        ;7848
                                50       04    AE    D0 0004E          MOVL    V_VALDESC, R0                           ;7849
                34   A4      14  A0             0C    29 00052          CMPC3   #12, 20(R0), 52(R4)
                                               03    12 00058          BNEQ    1$
                                           008D  31    0005A          BRW     6$
                                               56    DD 0005D 1$:      PUSHL   R6                                      ;7856
                                68             01    FB 0005F          CALLS   #1, DBG$PRINT
                                         28    A4    DD 00062          PUSHL   40(R4)                                  ;7857
                          00000000G  00          01    FB 00065          CALLS   #1, DBG$PRINT_IDENTIFIER
                                05             65    E9 0006C          BLBC    INVALID_FLAG, 2$                        ;7858
                                         25    A6    9F 0006F          PUSHAB  P.AEZ                                   ;7860
                                               03    11 00072          BRB     3$
                                         3C    A6    9F 00074 2$:      PUSHAB  P.AFA                                   ;7862
                                68             01    FB 00077 3$:      CALLS   #1, DBG$PRINT
                          00000000G  00          00    FB 0007A          CALLS   #0, DBG$NEWLINE                         ;7863
                                         10    AE    9F 00081          PUSHAB  DUMMY                                   ;7868
                                         0C    AE    9F 00084          PUSHAB  SYMID_LIST
                                         28    A4    DD 00087          PUSHL   40(R4)
                                69             03    FB 0008A          CALLS   #3, DBG$NGET_SYMID
                                0A             50    E9 0008D          BLBC    R0, 4$
                                         08    AE    DD 00090          PUSHL   SYMID_LIST                              ;7870
                          00000000G  00          01    FB 00093          CALLS   #1, DBG$STA_UNLOCK_SYMID
                                         10    AE    9F 0009A 4$:      PUSHAB  DUMMY                                   ;7871
                                         28    A4    DD 0009D          PUSHL   40(R4)
                          00000000G  00          02    FB 000A0          CALLS   #2, DBG$NFREE_DESC
                                7E       83    8F    9A 000A7          MOVZBL  #131, -(SP)                             ;7876
                                         34    A4    9F 000AB          PUSHAB  52(R4)
                          00000000G  00          02    FB 000AE          CALLS   #2, DBG$MAKE_VAL_DESC
                                04  AE           50    D0 000B5          MOVL    R0, V_VALDESC
04   BE           08  18 00000000G   00    F0 000B9          INSV    DBG$GB_LANGUAGE, #24, #8, @V_VALDESC            ;7877
                                         10    AE    9F 000C3          PUSHAB  DUMMY                                   ;7878
                                         0C    AE    9F 000C6          PUSHAB  SYMID_LIST
                                         0C    AE    DD 000C9          PUSHL   V_VALDESC
                                69             03    FB 000CC          CALLS   #3, DBG$NGET_SYMID
                                0A             50    E9 000CF          BLBC    R0, 5$
                                         08    AE    DD 000D2          PUSHL   SYMID_LIST                              ;7880
                          00000000G  00          01    FB 000D5          CALLS   #1, DBG$STA_LOCK_SYMID
                                               01    DD 000DC 5$:      PUSHL   #1                                      ;7881
                                         14    AE    9F 000DE          PUSHAB  DUMMY
```

```
                                28  A4  9F 000E1          PUSHAB  40(R4)                                    ;
                                10  AE  DD 000E4          PUSHL   V_VALDESC                               ;
                            67      04  FB 000E7          CALLS   #4, DBG$NCOPY_DESC                      ;
                                0C  AE  9F 000EA 68:      PUSHAB  VALDESC                                 ; 7893
                            7E  7A  8F  9A 000ED          MOVZBL  #122, -(SP)                             ;
                                0C  AE  DD 000F1          PUSHL   V_VALDESC                               ;
                  00000000G 00      03  FB 000F4          CALLS   #3, DBG$PRIM_TO_VAL                     ;
                                    01  DD 000FB          PUSHL   #1                                      ; 7894
                                14  AE  9F 000FD          PUSHAB  DUMMY                                   ;
                                04  A5  9F 00100          PUSHAB  NEWVALUE                                ;
                                18  AE  DD 00103          PUSHL   VALDESC                                 ;
                            67      04  FB 00106          CALLS   #4, DBG$NCOPY_DESC                      ;
                                    04     00109          RET                                             ; 7896

; Routine Size:  266 bytes,    Routine Base:  DBG$CODE + 2F23
```

```
;  7793          7897  1 ROUTINE PRIM_TO_VAL (PRIM_DESC, TARGET_TYPE, VAL_DESC) : NOVALUE =
;  7794          7898  1 !
;  7795          7899  1 ! FUNCTION
;  7796          7900  1 !       This routine is a cover routine for DBG$PRIM_TO_VAL. It
;  7797          7901  1 !       establishes a handler to catch error signals from DBG$PRIM_TO_VAL,
;  7798          7902  1 !       and then calls DBG$PRIM_TO_VAL to do the work.
;  7799          7903  1 !
;  7800          7904  1 ! INPUTS
;  7801          7905  1 !       See DBG$PRIM_TO_VAL in DBGVALUES.
;  7802          7906  1 !
;  7803          7907  1 ! OUTPUTS
;  7804          7908  1 !       See DBG$PRIM_TO_VAL in DBGVALUES.
;  7805          7909  1 !
;  7806          7910  2     BEGIN
;  7807          7911  2     ENABLE
;  7808          7912  2         PRIM_TO_VAL_HANDLER;
;  7809          7913  2     DBG$PRIM_TO_VAL (.PRIM_DESC, .TARGET_TYPE, .VAL_DESC);
;  7810          7914  1     END;


                                0000 00000 PRIM_TO_VAL:
                                           .WORD   Save nothing                    ;  7897
                        6D      0010   CF  DE 00002         MOVAL   1$, (FP)                ;  7910
                        7E        08   AC  7D 00007         MOVQ    TARGET_TYPE, -(SP)      ;  7913
                                  04   AC  DD 0000B         PUSHL   PRIM_DESC               ;
                  00000000G     00       03  FB 0000E         CALLS   #3, DBG$PRIM_TO_VAL     ;
                                             04 00015         RET                             ;  7914
                                0000 00016 1$:     .WORD   Save nothing                    ;  7910
                                           7E  D4 00018     CLRL    -(SP)                   ;
                                           5E  DD 0001A     PUSHL   SP                      ;
                              7E        04   AC  7D 0001C     MOVQ    4(AP), -(SP)            ;
                        0000V CF            03  FB 00020     CALLS   #3, PRIM_TO_VAL_HANDLER ;
                                             04 00025     RET                             ;

; Routine Size:  38 bytes,    Routine Base:  DBG$CODE + 302D
```

```
;  7812          7915  1 ROUTINE PRIM_TO_VAL_HANDLER(SIGARG, MECHARG, ENBLARG) =
;  7813          7916  1 !
;  7814          7917  1 ! FUNCTION
;  7815          7918  1 !       This is the error handler for the routine PRIM_TO_VAL. If
;  7816          7919  1 !       an error occurs during the evaluation of a Primary then this
;  7817          7920  1 !       routine is invoked and it sets the own variable INVALID_FLAG
;  7818          7921  1 !       to indicate that the Primary is now invalid.
;  7819          7922  1 !
;  7820          7923  1 ! INPUTS
;  7821          7924  1 !       SIGARG  - The signal argument vector.
;  7822          7925  1 !
;  7823          7926  1 !       MECHARG - The mechanism argument vector.
;  7824          7927  1 !
;  7825          7928  1 !       ENBLARG - The enable argument vector (EVENT pointer is passed in
;  7826          7929  1 !                 from here).
;  7827          7930  1 !
;  7828          7931  1 ! OUTPUT
;  7829          7932  1 !       The OWN variable INVALID_FLAG is set to TRUE.
;  7830          7933  1 !
;  7831          7934  2     BEGIN
;  7832          7935  2     MAP
;  7833          7936  2         ENBLARG: REF VECTOR[,LONG],
;  7834          7937  2         SIGARG: REF VECTOR[,LONG];
;  7835          7938  2
;  7836          7939  2
;  7837          7940  2     ! If we were unwinding, continue unwinding.
;  7838          7941  2     !
;  7839          7942  2     IF .SIGARG[1] EQL SS$_UNWIND
;  7840          7943  2     THEN
;  7841          7944  2         RETURN SS$_CONTINUE;
;  7842          7945  2
;  7843          7946  2
;  7844          7947  2     ! Set the OWN flag saying the Primary is invalid.
;  7845          7948  2     !
;  7846          7949  2     INVALID_FLAG = TRUE;
;  7847          7950  2
;  7848          7951  2
;  7849          7952  2     ! Unwind to the caller of PRIM_TO_VAL.
;  7850          7953  2     !
;  7851          7954  2     $SETUNWIND();
;  7852          7955  2     RETURN 0;
;  7853          7956  1     END;


                                0000 00000 PRIM_TO_VAL_HANDLER:
                                           .WORD   Save nothing                    ;  7915
                     50    04   AC  D0 00002          MOVL    SIGARG, R0          ;  7942
00000920             8F    04   A0  D1 00006          CMPL    4(R0), #2336
                                04  12 0000E          BNEQ    1$
                     50         01  D0 00010          MOVL    #1, R0              ;  7944
                                04  00013          1$: RET
00000000'            EF         01  D0 00014          MOVL    #1, INVALID_FLAG    ;  7949
                                7E  7C 0001B          CLRQ    -(SP)               ;  7954
00000000G            00         02  FB 0001D          CALLS   #2, SYS$UNWIND
```

```
                50  D4 00024          CLRL    R0                              ; 7955
                    04 00026          RET                                     ; 7956

; Routine Size:  39 bytes,    Routine Base:  DBG$CODE + 3053
```

```
;  7855          7957  1 ROUTINE PARSE_WHEN_CONDITION(EVENT): NOVALUE =
;  7856          7958  1 !
;  7857          7959  1 ! FUNCTION
;  7858          7960  1 !       This routine calls the parser to parse the when condition.
;  7859          7961  1 !       If any error occurs during the parsing, final handler catches
;  7860          7962  1 !       the error message, announce the message, WHEN_CONDITION is set
;  7861          7963  1 !       to FALSE in this case, Break point remains set.
;  7862          7964  1 !
;  7863          7965  1 ! INPUTS
;  7864          7966  1 !       EVENT   - The address of the event_entry.
;  7865          7967  1 !
;  7866          7968  1 ! OUTPUTS
;  7867          7969  1 !       None.
;  7868          7970  1 !
;  7869          7971  1
;  7870          7972  2     BEGIN
;  7871          7973  2
;  7872          7974  2     MAP
;  7873          7975  2         EVENT: REF EVENT$EVENT_DESCRIPTOR;  ! Event entry pointer
;  7874          7976  2
;  7875          7977  2     BUILTIN CALLG;
;  7876          7978  2
;  7877          7979  2     LOCAL
;  7878          7980  2         CONDITION,                          ! local when condition value
;  7879          7981  2         CONDITION_VAL :     DBG$STG_DESC,   ! Condition Value Descr
;  7880          7982  2         CONDITION_STR :     DBG$STG_DESC,   ! Condition String Descr
;  7881          7983  2         CONDITION_TXT :     REF VECTOR [, WORD], ! Condition text
;  7882          7984  2         CONDITION_RDX,                      !    radix
;  7883          7985  2         MESSAGE_VECT,                       ! Message Vector
;  7884          7986  2         WHEN_ENTRY :    REF EVENT$WHEN_DESCRIPTOR;  ! WHEN Descriptor
;  7885          7987  2
;  7886          7988  2     ENABLE
;  7887          7989  2         DBG$FINAL_HANDL;
;  7888          7990  2
;  7889          7991  2
;  7890          7992  2     ! Assume WHEN condition is always true to begin with, so just in
;  7891          7993  2     ! case, bad things happened during the calls to other routine,
;  7892          7994  2     ! break point is still taken.
;  7893          7995  2     !
;  7894          7996  2     WHEN_CONDITION = TRUE;
;  7895          7997  2
;  7896          7998  2
;  7897          7999  2     ! Point to the WHEN entry.
;  7898          8000  2     !
;  7899          8001  2     WHEN_ENTRY = .EVENT [EVENT$L_WHEN];
;  7900          8002  2
;  7901          8003  2
;  7902          8004  2     ! Point to the WHEN expression.
;  7903          8005  2     !
;  7904          8006  2     CONDITION_TXT = .WHEN_ENTRY [EVENT$L_WHEN_POINT];
;  7905          8007  2
;  7906          8008  2
;  7907          8009  2     ! Set up the condition descriptors.
;  7908          8010  2     !
;  7909          8011  2     CONDITION_STR [DSC$B_CLASS] = DSC$K_CLASS_S;
;  7910          8012  2     CONDITION_STR [DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  7911          8013  2     CONDITION_STR [DSC$W_LENGTH] = .CONDITION_TXT [0];
```

```
;  7912     8014  2     CONDITION_STR [DSC$A_POINTER] = CONDITION_TXT [1];
;  7913     8015
;  7914     8016        ! Set up some globals that C uses for purposes of preseving the
;  7915     8017        ! original casing.
;  7916     8018        !
;  7917     8019        IF .DBG$GB_LANGUAGE EQL DBG$K_C
;  7918     8020        THEN
;  7919     8021            BEGIN
;  7920     8022            DBG$GL_ORIG_COMMAND_PTR = CONDITION_TXT [1];
;  7921     8023            DBG$GL_UPCASE_COMMAND_PTR[0] = CONDITION_TXT [1];
;  7922     8024            DBG$GL_UPCASE_COMMAND_PTR[1] = CONDITION_TXT[1] + .CONDITION_TXT[0] - 1;
;  7923     8025            END;
;  7924     8026
;  7925     8027        ! Determine the current radix.
;  7926     8028        !
;  7927     8029        CONDITION_RDX = .DBG$GB_RADIX[DBG$B_RADIX_INPUT];
;  7928     8030
;  7929     8031
;  7930     8032        ! Obtain a value descriptor for the condition.
;  7931     8033        !
;  7932     8034        IF NOT DBG$NPARSE_EXPRESSION
;  7933     8035                    (CONDITION_STR,
;  7934     8036                     .CONDITION_RDX,
;  7935     8037                     CONDITION,
;  7936     8038                     TOKEN$K_TERM_NONE,
;  7937     8039                     MESSAGE_VECT
;  7938     8040                    )
;  7939     8041        THEN
;  7940     8042            CALLG (.MESSAGE_VECT, LIB$SIGNAL);
;  7941     8043
;  7942     8044
;  7943     8045        ! Set up the vax descriptor for the condition.
;  7944     8046        ! For now, we just declare the descriptor to be longword integer,
;  7945     8047        ! since this causes the fewest problems in the type converter.
;  7946     8048        ! Eventually, if we get a Boolean type and all languages support
;  7947     8049        ! it then we will build a target descriptor of this type.
;  7948     8050        !
;  7949     8051        CONDITION_VAL [DSC$B_CLASS] = DSC$K_CLASS_S;
;  7950     8052        CONDITION_VAL [DSC$B_DTYPE] = DSC$K_DTYPE_L;
;  7951     8053        CONDITION_VAL [DSC$W_LENGTH] = 4;
;  7952     8054        CONDITION_VAL [DSC$A_POINTER] = WHEN_CONDITION;
;  7953     8055        CONDITION_VAL [DSC$L_POS] = 0;
;  7954     8056
;  7955     8057
;  7956     8058        ! Special case for PASCAL. PASCAL returns descriptors
;  7957     8059        ! of type Boolean (dsc$k_dtype_tf) for relational expressions.
;  7958     8060        !
;  7959     8061        IF .DBG$GB_LANGUAGE EQL DBG$K_PASCAL
;  7960     8062        THEN
;  7961     8063            BEGIN
;  7962     8064            CONDITION_VAL [DSC$B_DTYPE] = DSC$K_DTYPE_TF;
;  7963     8065            CONDITION_VAL [DSC$W_LENGTH] = 1;
;  7964     8066            END;
;  7965     8067
;  7966     8068
;  7967     8069        ! Do the conversion from value descriptor to integer.  NOTE THAT
;  7968     8070        ! I'M REUSING THE DESCRIPTOR, AND CONDITION LONGWORD.
```

```
;  7969          8071  2     !
;  7970          8072  2     IF NOT DBG$NTYPE_CONV (.CONDITION,
;  7971          8073  2                            DBG$K_DEFAULT,
;  7972          8074  2                            DBG$K_VAX_DESC,
;  7973          8075  2                            CONDITION_VAL,
;  7974          8076  2                            MESSAGE_VECT
;  7975          8077  2                           )
;  7976          8078  2     THEN
;  7977          8079  2         CALLG (.MESSAGE_VECT, LIB$SIGNAL);
;  7978          8080  2
;  7979          8081  2     RETURN 0;
;  7980          8082  1     END;


                                003C 00000 PARSE_WHEN_CONDITION:
                                           .WORD   Save R2,R3,R4,R5                                        ; 7957
                 55 00000000'   EF  9E 00002       MOVAB   WHEN_CONDITION, R5                              ;
                 54 00000000G   00  9E 00009       MOVAB   LIB$SIGNAL, R4                                  ;
                 53 00000000G   00  9E 00010       MOVAB   DBG$GB_LANGUAGE, R3                             ;
                 5E             20  C2 00017       SUBL2   #32, SP                                         ;
                 6D      009D   CF  DE 0001A       MOVAL   5$, (FP)                                        ; 7972
                 65             01  D0 0001F       MOVL    #1, WHEN_CONDITION                              ; 7996
                 50       04    AC  D0 00022       MOVL    EVENT, R0                                       ; 8001
                 50       20    A0  D0 00026       MOVL    32(R0), WHEN_ENTRY                              ;
                 50       0C    A0  D0 0002A       MOVL    12(WHEN_ENTRY), CONDITION_TXT                   ; 8006
              0A AE       010E  8F  B0 0002E       MOVW    #270, CONDITION_STR+2                           ; 8012
                 52             60  3C 00034       MOVZWL  (CONDITION_TXT), R2                             ; 8013
              08 AE             52  B0 00037       MOVW    R2, CONDITION_STR                               ;
                 51       02    A0  9E 0003B       MOVAB   2(R0), R1                                       ; 8014
              0C AE             51  D0 0003F       MOVL    R1, CONDITION_STR+4                             ;
                 07             63  91 00043       CMPB    DBG$GB_LANGUAGE, #7                             ; 8019
                                17  12 00046       BNEQ    1$                                              ;
 00000000G       00             51  D0 00048       MOVL    R1, DBG$GL_ORIG_COMMAND_PTR                     ; 8022
 00000000G       00             51  D0 0004F       MOVL    R1, DBG$GL_UPCASE_COMMAND_PTR                   ; 8023
 00000000G       00       01 A240  9E 00056       MOVAB   1(R2)[CONDITION_TXT], -                         ; 8024
                                                          DBG$GL_UPCASE_COMMAND_PTR+4
                 50 00000000G   00  9A 0005F 1$:   MOVZBL  DBG$GB_RADIX, CONDITION_RDX                     ; 8029
                          04    AE  9F 00066       PUSHAB  MESSAGE_VECT                                    ; 8035
                                7E  D4 00069       CLRL    -(SP)                                           ;
                          08    AE  9F 0006B       PUSHAB  CONDITION                                       ;
                                50  DD 0006E       PUSHL   CONDITION_RDX                                   ; 8036
                          18    AE  9F 00070       PUSHAB  CONDITION_STR                                   ; 8035
 00000000G       00             05  FB 00073       CALLS   #5, DBG$NPARSE_EXPRESSION                       ;
                 04             50  E8 0007A       BLBS    R0, 2$                                          ;
                 64       04    BE  FA 0007D       CALLG   @MESSAGE_VECT, LIB$SIGNAL                       ; 8042
              14 AE 01080004    8F  D0 00081 2$:   MOVL    #17301508, CONDITION_VAL                        ; 8053
              18 AE             65  9E 00089       MOVAB   WHEN_CONDITION, CONDITION_VAL+4                 ; 8054
                          1C    AE  D4 0008D       CLRL    CONDITION_VAL+8                                 ; 8055
                 06             63  91 00090       CMPB    DBG$GB_LANGUAGE, #6                             ; 8061
                                08  12 00093       BNEQ    3$                                              ;
              16 AE             28  90 00095       MOVB    #40, CONDITION_VAL+2                            ; 8064
              14 AE             01  B0 00099       MOVW    #1, CONDITION_VAL                               ; 8065
                          04    AE  9F 0009D 3$:   PUSHAB  MESSAGE_VECT                                    ; 8072
                          18    AE  9F 000A0       PUSHAB  CONDITION_VAL                                   ;
```

```
                          7E      82    8F   9A 000A3          MOVZBL  #130, -(SP)                                    ;
                                        01   DD 000A7          PUSHL   #1                                              ;
                                  10    AE   DD 000A9          PUSHL   CONDITION                                       ;
                00000000G   00          05   FB 000AC          CALLS   #5, DBG$NTYPE_CONV                              ;
                            04          50   E8 000B3          BLBS    R0, 4$                                          ;
                            64    04    BE   FA 000B6          CALLG   @MESSAGE_VECT, LIB$SIGNAL                       ; 8079
                                        04   00 000BA 4$:       RET                                                     ; 8082
                                      0000   000BB 5$:         .WORD   Save nothing                                    ; 7972
                                   7E   D4   000BD             CLRL    -(SP)                                           ;
                                   5E   DD   000BF             PUSHL   SP                                              ;
                          7E        04   AC   7D 000C1         MOVQ    4(AP), -(SP)                                    ;
                00000000G   00          03   FB 000C5          CALLS   #3, DBG$FINAL_HANDL                             ;
                                        04   00 000CC          RET                                                     ;

; Routine Size:  205 bytes,    Routine Base:  DBG$CODE + 307A

; 7981          8083  1
```

```
;  7983          8084  1 ROUTINE MATCH_RIGHT_CALL_FRAME(EVENT) =
;  7984          8085  1 !
;  7985          8086  1 ! FUNCTION
;  7986          8087  1 !       This routine is used to match the exact call frame at the time
;  7987          8088  1 !       we set bit 15, or turn off bit 15 in saved PSW and the time we got
;  7988          8089  1 !       reserved operand fault.
;  7989          8090  1 !
;  7990          8091  1 !       For FP pointer may be changed inside of the call by the called
;  7991          8092  1 !       routine.  So to be safe, we don't just compare the pointers, we
;  7992          8093  1 !       check some call frame contents as well.
;  7993          8094  1 !
;  7994          8095  1 ! INPUTS
;  7995          8096  1 !       Event entry for the event we consister.
;  7996          8097  1 !
;  7997          8098  1 ! OUTPUTS
;  7998          8099  1 !       True for Yes, this is the frame caused the event,
;  7999          8100  1 !       Fasle for No, this is not the frame caused the event.
;  8000          8101  1 !
;  8001          8102  1
;  8002          8103  2     BEGIN
;  8003          8104  2     MAP
;  8004          8105  2         EVENT: REF EVENT$EVENT_DESCRIPTOR;
;  8005          8106  2
;  8006          8107  2     LOCAL
;  8007          8108  2         CALL_FRAME: REF BLOCK[,BYTE],   ! Pointer to FP
;  8008          8109  2         CALL_MASK: WORD,                ! Mask in Call frame
;  8009          8110  2         RUN_FRAME: REF BLOCK[,BYTE],    ! FP was saved in DEBUG run frame
;  8010          8111  2         RUN_MASK: WORD;                 ! Mask in current FP
;  8011          8112  2
;  8012          8113  2
;  8013          8114  2     ! If there is no FP existed, simply returns.
;  8014          8115  2     !
;  8015          8116  2     IF .EVENT [EVENT$L_USERS_FP] EQL 0 THEN RETURN FALSE;
;  8016          8117  2
;  8017          8118  2     ! Dig out current FP from DEBUG runframe.
;  8018          8119  2     !
;  8019          8120  2     RUN_FRAME = .DBG$RUNFRAME [DBG$L_USER_FP];
;  8020          8121  2
;  8021          8122  2     ! If the call frame saved in event entry is the same as current FP,
;  8022          8123  2     ! got it, no problem.
;  8023          8124  2     !
;  8024          8125  2     IF .EVENT [EVENT$L_USERS_FP] EQL .RUN_FRAME
;  8025          8126  2     THEN
;  8026          8127  2         RETURN TRUE;
;  8027          8128  2
;  8028          8129  2     ! Need to do further investigation.  Dig out the contents of the call
;  8029          8130  2     ! frame was saved from Call frame.
;  8030          8131  2     !
;  8031          8132  2     CALL_FRAME = .EVENT [EVENT$L_CALL_FRAME];
;  8032          8133  2     IF .CALL_FRAME EQL 0 THEN RETURN FALSE;
;  8033          8134  2
;  8034          8135  2     ! Compare the mask.  But don't compare the stack alignment, compare
;  8035          8136  2     ! register mask and calls flag.  Compare saved AP, and FP.
;  8036          8137  2     !
;  8037          8138  2     CALL_MASK = .CALL_FRAME[SF$W_SAVE_MASK];
;  8038          8139  2     RUN_MASK = .RUN_FRAME[SF$W_SAVE_MASK];
;  8039          8140  2     CALL_MASK = .CALL_MASK AND %X'7FFF';
```

```
; 8040          8141  2      RUN_MASK = .RUN_MASK AND %X'7FFF';
; 8041          8142  2      IF (.CALL_MASK EQL .RUN_MASK) AND
; 8042          8143  2         (.CALL_FRAME[SF$L_SAVE_AP] EQL .RUN_FRAME[SF$L_SAVE_AP]) AND
; 8043          8144  3         (.CALL_FRAME[SF$L_SAVE_FP] EQL .RUN_FRAME[SF$L_SAVE_FP])
; 8044          8145  2      THEN
; 8045          8146  3          BEGIN
; 8046          8147  3
; 8047          8148  3          ! Got it, update the FP to current FP.  So we can set/clear bit 15
; 8048          8149  3          ! in saved PSW in the correct FP.
; 8049          8150  3          !
; 8050          8151  3          EVENT [EVENT$L_USERS_FP] = .RUN_FRAME;
; 8051          8152  3          RETURN TRUE;
; 8052          8153  3          END
; 8053          8154  2      ELSE
; 8054          8155  2          RETURN FALSE;
; 8055          8156  2
; 8056          8157  1      END;


                                003C 00000 MATCH_RIGHT_CALL_FRAME:
                                                  .WORD    Save R2,R3,R4,R5                          ; 8084
                 50        04   AC  D0 00002      MOVL     EVENT, R0                                 ; 8116
                           24   A0  D5 00006      TSTL     36(R0)                                    ;
                                46  13 00009      BEQL     2$                                        ;
                 51  00000000G  00  D0 0000B      MOVL     DBG$RUNFRAME+56, RUN_FRAME                ; 8120
                 51        24   A0  D1 00012      CMPL     36(R0), RUN_FRAME                         ; 8125
                                35  13 00016      BEQL     1$                                        ;
                 52        3C   A0  D0 00018      MOVL     60(R0), CALL_FRAME                        ; 8132
                                33  13 0001C      BEQL     2$                                        ; 8133
                 54        06   A2  B0 0001E      MOVW     6(CALL_FRAME), CALL_MASK                  ; 8138
                 53        06   A1  B0 00022      MOVW     6(RUN_FRAME), RUN_MASK                    ; 8139
    55             54       0F  00  EF 00026      EXTZV    #0, #15, CALL_MASK, R5                    ; 8140
                           54   55  B0 0002B      MOVW     R5, CALL_MASK
    55             53       0F  00  EF 0002E      EXTZV    #0, #15, RUN_MASK, R5                     ; 8141
                           53   55  B0 00033      MOVW     R5, RUN_MASK
                           53   54  B1 00036      CMPW     CALL_MASK, RUN_MASK                       ; 8142
                                16  12 00039      BNEQ     2$                                        ;
              08   A1      08   A2  D1 0003B      CMPL     8(CALL_FRAME), 8(RUN_FRAME)               ; 8143
                                0F  12 00040      BNEQ     2$                                        ;
              0C   A1      0C   A2  D1 00042      CMPL     12(CALL_FRAME), 12(RUN_FRAME)             ; 8144
                                08  12 00047      BNEQ     2$                                        ;
              24   A0           51  D0 00049      MOVL     RUN_FRAME, 36(R0)                         ; 8151
                   50           01  D0 0004D 1$:  MOVL     #1, R0                                    ; 8155
                                    04 00050      RET                                                ;
                                50  D4 00051 2$:  CLRL     R0                                        ; 8157
                                    04 00053      RET                                                ;

; Routine Size:  84 bytes,    Routine Base:  DBG$CODE + 3147


; 8057          8158  1
; 8058          8159  1 END                               ! End of module
; 8059          8160  1
; 8060          8161  0 ELUDOM
```

```
                                                .EXTRN  LIB$SIGNAL, SYS$UNWIND

;                               PSECT SUMMARY
;
;       Name                    Bytes                   Attributes
;
;  DBG$GLOBAL                     69  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  DBG$OWN                       356  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  DBG$CODE                    12699  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)
;  DBG$PLIT                     1450  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)


;                           Library Statistics
;                                    --------  Symbols  --------    Pages       Processing
;       File                         Total     Loaded   Percent     Mapped      Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1      18619        42         0       1000      00:01.8
;  _$255$DUA28:[DEBUG.OBJ]STRUCDEF.L32;1
;                                        32         2         6          7      00:00.2
;  _$255$DUA28:[DEBUG.OBJ]DBGLIB.L32;1
;                                      1545       267        17         97      00:02.0
;  _$255$DUA28:[DEBUG.OBJ]DSTRECRDS.L32;1
;                                       418       106        25         31      00:00.4
;  _$255$DUA28:[DEBUG.OBJ]DBGMSG.L32;1
;                                       386        26         6         22      00:00.3
;  _$255$DUA28:[DEBUG.OBJ]DBGGEN.L32;1
;                                       150         7         4         12      00:00.3


;                                   COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:DBGEVENT/OBJ=OBJ$:DBGEVENT MSRC$:DBGEVENT/UPDATE=(ENH$:DBGEVENT)

; Size:         12699 code + 1875 data bytes
; Run Time:         04:04.0
; Elapsed Time:     13:38.6
; Lines/CPU Min:     2006
; Lexemes/CPU-Min: 14391
; Memory Used:  1029 pages
; Compilation Complete
```

DBGEVENT
.LIS

DBGEXT
LIS

DBGEXADEP
LIS

DBGEXC
LIS